The Undocumented Z80 Documented

Sean Young

Version 0.91, 18th September, 2005

Copyright Statement

Contents

Chapter 1

Introduction

1.1 History

(Sean) Ever since I first started working on an MSX emulator, I've been very interested in getting the emulation absolutely correct — including the undocumented features. Not just to make sure that all games work, but also to make sure that if a program crashes, it crashes exactly the same way if running on an emulator as on the real thing. Only then is perfection achieved.

I set about collecting information. I found pieces of information on the Internet, but not everything there is to know. So I tried to fill in the gaps, the results of which I put on my website. Various people have helped since then; this is the result of all those efforts and to my knowledge this document is the most complete.

(Jan) Interested in emulation for a long time, but a few years after Sean started writing this document, I have also started writing my own MSX emulator in 2003 and I've used this document quite a lot. Now (2005) the Z80 emulation is nearing perfection, I decided to add what extra I have learned and comments various people have sent to Sean, to this document.

I have restyled the document (although very little) to fit my personal needs and I have checked a lot of things that were already in here.

1.2 Where to get this document

The latest version is always available in LaTeX and pdf at the following location:
`http://www.myquest.nl/z80undocumented/`

1.3 Feedback

I welcome any kind of feedback. I would like to hear about any corrections or additions you might have. Also note that there are a few flags which are still unknown, it would be great if someone found out how they work. You can reach me at `jw@dds.nl` and my website can be found at `http://www.myquest.nl/z80undocumented/`. Sean's website is at `http://www.msxnet.org/`.

1.4 ChangeLog

18th September 2005 (version 0.91) Corrected a textual typo in the R register and memory refresh section, thanks to David Aubespin. Corrected the contradiction in the DAA section saying the NF flag was both affected and unchanged :) thanks to Dan Meir. Added an error in official documetation about that way Interrupt Mode 2 works, thanks to Aaldert Dekker.

15th Juni 2005 (version 0.9) Corrected improper notation of JP x,nn mnemonics in opcode list, thanks to Laurens Holst. Corrected a mistake in the INI, INIR, IND, INDR section and documented a mistake in official Z80 documentation concerning Interrupt Mode 2, thanks to Boris Donko. Thanks to Aaldert Dekker for his ideas, for verifying many assumptions and writing instruction exercisers for various instruction groups.

18th May 2005 (version 0.8) Added an alphabetical list of instructions for easy reference and corrected an error in the 16-bit arithmetic section, SBC HL, nn sets the N-flag just like other subtraction instructions, thanks to Fredrik Olsson for pointing that out.

4th April 2005 (version 0.7) I (Jan <jw@dds.nl>) will be maintaining this document from this version on. I restyled the document to fix the page numbering issues, corrected an error in the I/O Block Instructions section, added graphics for the RLD and RRD instructions and corrected the spelling in several places.

20th November 2003 (version 0.6) Again, thanks to Ramsoft, added PF flag to `OUTI`, `INI` and friends. Minor fix to `DAA` tables, other minor fixes.

13th November 2003 (version 0.5) Thanks to Ramsoft, add the correct tables for the `DAA` instruction (section 4.7). Minor corrections & typos, thanks to Jim Battle, David Sutherland and most of all Fred Limouzin.

September 2001 (version 0.4) Previous documents I had written were in plain text and Microsoft Word, which I now find very embarrassing, so I decided to combine them all and use LaTeX. Apart from a full re-write, the only changed information is “Power on defaults” (section 2.4) and the algorithm for the CF and HF flags for `OTIR` and friends (section 4.3).

Chapter 2

Overview

2.1 History of the Z80

In 1969 Intel was approached by a Japanese company called Busicom to produce chips for Busicom's electronic desktop calculator. Intel suggested that the calculator should be built around a single-chip generalized computing engine and thus was born the first microprocessor — the 4004. Although it was based on ideas from much larger mainframe and mini-computers the 4004 was cut down to fit onto a 16-pin chip, the largest that was available at the time, so that its data bus and address bus were each only 4-bits wide.

Intel went on to improve the design and produced the 4040 (an improved 4-bit design) the 8008 (the first 8-bit microprocessor) and then in 1974 the 8080. This last one turned out to be a very useful and popular design and was used in the first home computer, the Altair 8800, and CP/M.

In 1975 Federico Faggin who had had worked at Intel on the 4004 and its successors left the company and joined forces with Masatoshi Shima to from Zilog. At their new company Faggin and Shima designed a microprocessor that was compatible with Intel's 8080 (it ran all 78 instructions of the 8080 in almost the same way that Intel's chip did)[1] but had many more abilities (an extra 120 instructions, many more registers, simplified connection to hardware). Thus was born the mighty Z80! and thus was the empire forged.

The original Z80 was first released in July 1976, coincidentally Jan was born in the very same month. Since then newer versions have appeared with much of the same architecture but running at higher speeds. The original Z80 ran with a clock rate of 2.5MHz, the Z80A runs at 4MHz, the Z80B at 6MHz and the Z80H at 8Mhz.

Many companies produced machines based around Zilog's improved chip during the 1970's and 80's and because the chip could run 8080 code without needing any changes to the code the perfect choice of operating system was CP/M.

Also Zilog has created a Z280, an enhanced version of the Zilog Z80 with a 16 bit architecture, introduced in July, 1987. It added an MMU to expand addressing to 16Mb, features for multitasking, a 256 byte cache, and a huge number of new opcodes (giving a total of over 2000!). Its internal clock runs at 2 or 4 times the external clock (e.g. a 16MHz CPU with a 4MHz bus

The Z380 CPU incorporates advanced architectural while maintaining Z80/ Z180 object code compatibility. The Z380 CPU is an enhanced version of the Z80 CPU. The Z80 instruction set has been retained, adding a full compliment of 16-bit arithmetic and logical operations, multiply and divide, a complete set of register-to-register loads and exchanges, plus 32-bit load and exchange, and 32-bit arithmetic operations for address calculations.

The addressing modes of the Z80 have been enhanced with Stack pointer relative loads and stores, 16-bit and 24- bit indexed offsets and more flexible indirect register addressing. All of the

[1] Thanks to Jim Battle <frustum@pacbell.net>: the 8080 always puts the parity in the PF flag; VF does not exist and the timing is different. Possibly there are other differences.

addressing modes allow access to the entire 32-bit addressing space.

2.2 Registers

The following accessible registers exist in the Z80.

Register		Description
A	F	Accumulator and Flags
BC		
DE		General purpose registers
HL		
IX		Index registers
IY		
PC		
SP		Special purpose registers
I	R	
AF'		
BC'		Alternate general purpose registers
DE'		
HL'		

For interrupts, there are two interrupt flop-flops, IFF1 and IFF2, and the interrupt mode is retained. See chapter 5 for more about interrupts. Also there is an internal register which is described in section 4.3.

2.3 Flags

The conventional way of denoting the flags is with one letter, 'C' for the carry flag for example. It could be confused with the C register, so I've chosen to use the 'CF' notation for flags. Also in previous things I've written I called the two undocumented flags 5 and 3, but now I've changed to the same notation used in MAME[2], which is YF and XF, respectively. Note that in mnemonics the original way is still maintained.

bit	7	6	5	4	3	2	1	0
flag	SF	ZF	YF	HF	XF	PF	NF	CF

SF flag Set if the 2-complement value is negative. It's simply a copy of the most significant bit.

ZF flag Set if the result is zero.

YF flag A copy of bit 5 of the result.

HF flag The half-carry of an addition/subtraction (from bit 3 to 4). Needed for BCD correction with `DAA`.

XF flag A copy of bit 3 of the result.

PF flag This flag can either be the parity of the result (PF), or the 2-compliment signed overflow (VF): set if 2-compliment value doesn't fit in the register.

NF flag Shows whether the last operation was an addition (0) or an subtraction (1). This information is needed for `DAA`.[3]

[2] http://www.mame.net/

[3] Wouldn't it be better to have separate instructions for `DAA` after addition and subtraction, like the 80x86 has in stead of sacrificing a bit in the flag register?

CF flag The carry flag, set if there was a carry after the most significant bit.

Note that the only way to read the XF, YF and NF can only be read using `PUSH AF`.

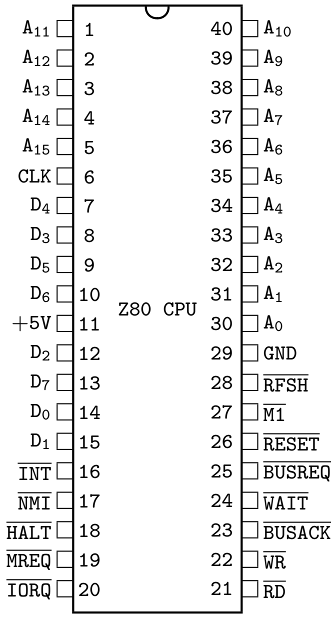
2.4 Power on defaults

Matt[4] has done some excellent research on this. He found that AF and SP are always set to FFFFh after a reset, and all other registers are undefined (different depending on how long the CPU has been powered off, different for different Z80 chips). Of course the PC should be set to 0 after a reset, and so should the IFF1 and IFF2 flags (otherwise strange things could happen). Also since the Z80 is 8080 compatible, interrupt mode is probably 0.

Probably the best way to simulate this in an emulator is set PC, IFF1, IFF2, IM to 0 and set all other registers to FFFFh.

2.5 Pin Descriptions [7]

This section might also relevant even if you don't do anything with hardware; it might give so insight into how the Z80 operates. Besides, it took me hours to draw this.



$\mathtt{A}_{15} - \mathtt{A}_0$ *Address bus* (output, active high, 3-state). This bus is used for accessing the memory and for I/O ports. During the refresh cycle the IR register is put on this bus.

$\overline{\mathtt{BUSACK}}$ *Bus Acknowledge* (output, active low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{\mathtt{MREQ}}$, $\overline{\mathtt{IORQ}}$, $\overline{\mathtt{RD}}$ and $\overline{\mathtt{WR}}$ have been entered into their high-impedance states. The external device now control these lines.

$\overline{\mathtt{BUSREQ}}$ *Bus Request* (input, active low). Bus Request has a higher priority than $\overline{\mathtt{NMI}}$ and is always recognised at the end of the current machine cycle. $\overline{\mathtt{BUSREQ}}$ forces the CPU address bus, data bus and control signals $\overline{\mathtt{MREQ}}$, $\overline{\mathtt{IORQ}}$, $\overline{\mathtt{RD}}$ and $\overline{\mathtt{WR}}$ to go to a high-impedance state so that other devices can control these lines. $\overline{\mathtt{BUSREQ}}$ is normally wired-OR and requires an external pullup for these applications. Extended $\overline{\mathtt{BUSREQ}}$ periods due to extensive DMA operations can prevent the CPU from refreshing dynamic RAMs.

[4] redflame@xmission.com

$\mathtt{D}_7 - \mathtt{D}_0$ *Data Bus* (input/output, active low, 3-state). Used for data exchanges with memory, I/O and interrupts.

$\overline{\mathtt{HALT}}$ *Halt State* (output, active low). Indicates that the CPU has executed a `HALT` instruction and is waiting for either a maskable or nonmaskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU stops increasing the PC so the instruction is re-executed, to maintain memory refresh.

$\overline{\mathtt{INT}}$ *Interrupt Request* (input, active low). Interrupt Request is generated by I/O devices. The CPU honours a request at the end of the current instruction if IFF1 is set. $\overline{\mathtt{INT}}$ is normally wired-OR and requires an external pullup for these applications.

$\overline{\mathtt{IORQ}}$ *Input/Output Request* (output, active low, 3-state). Indicates that the address bus holds a vailid I/O address for an I/O read or write operation. $\overline{\mathtt{IORQ}}$ is also generated concurrently with $\overline{\mathtt{M1}}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the databus.

$\overline{\mathtt{M1}}$ *Machine Cycle One* (output, active low). $\overline{\mathtt{M1}}$, together with $\overline{\mathtt{MREQ}}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{\mathtt{M1}}$, together with $\overline{\mathtt{IORQ}}$, indicates an interrupt acknowledge cycle.

$\overline{\mathtt{MREQ}}$ *Memory Request* (output, active low, 3-state). Indicates that the address holds a valid address for a memory read or write cycle operations.

$\overline{\mathtt{NMI}}$ *Non-Maskable Interrupt* (input, negative edge-triggered). $\overline{\mathtt{NMI}}$ has a higher priority than $\overline{\mathtt{INT}}$. $\overline{\mathtt{NMI}}$ is always recognised at the end of an instruction, independent of the status of the interrupt flip-flops and automatically forces the CPU to restart at location 0066h.

$\overline{\mathtt{RD}}$ *Read* (output, active low, 3-state). Indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to place data onto the data bus.

$\overline{\mathtt{RESET}}$ *Reset* (input, active low). Initializes the CPU as follows: it resets the interrupt flip-flops, clears the PC and IR registes, and set the interrupt mode to 0. During reset time, the address bus and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{\mathtt{RESET}}$ must be active for a minimum of three full clock cycles before the reset operation is complete. Note that Matt found that SP and AF are set to FFFFh.

$\overline{\mathtt{RFSH}}$ *Refresh* (output, active low). $\overline{\mathtt{RFSH}}$, together with $\overline{\mathtt{MREQ}}$, indicates that the IR registers are on the address bus (note that only the lower 7 bits are useful) and can be used for the refresh of dynamic memories.

$\overline{\mathtt{WAIT}}$ *Wait* (input, active low). Indicates to the CPU that the addressed memory or I/O device are not ready for data transfer. The CPU continues to enter a wait state as long as this signal is active. Note that during this period memory is not refreshed.

$\overline{\mathtt{WR}}$ *Write* (output, active low, 3-state). Indicates that the CPU wants to write data to memory or an I/O device. The addressed I/O device or memory should use this signal to store the data on the data bus.

Chapter 3

Undocumented Opcodes

There are quite a few undocumented opcodes/instructions. This section should describe every possible opcode so you know what will be executed, whatever the value of the opcode is.

The following prefixes exist: CB, ED, DD, FD, DDCB and FDCB. Prefixes change the way the following opcodes are interpreted. All instructions without a prefix (not a value of one the above) are single byte opcodes[1], which are documented in the official documentation.

3.1 CB Prefix [5]

An opcode with a CB prefix is a rotate, shift or bit test/set/reset instruction. There are a few instructions missing from the official list, which are usually denoted with `SLL` (Shift Logical Left). It works like `SLA`, for one exception: it sets bit 0 (`SLA` resets it).

```
CB30    SLL B
CB31    SLL C
CB32    SLL D
CB33    SLL E
CB34    SLL H
CB35    SLL L
CB36    SLL (HL)
CB37    SLL A
```

3.2 DD Prefix [5]

In general, the instruction following the DD prefix is executed as is, but if the HL register is supposed to be used the IX register is used instead. Here are the rules:

- Any usage of HL is treated as an access to IX (except `EX DE,HL` and `EXX` and the ED prefixed instructions that use HL).
- Any access to (HL) is changed to (IX+d), where 'd' is a signed displacement byte placed after the main opcode — except `JP (HL)`, which isn't indirect anyway. The mnemonic should be `JP HL`.
- Any access to H is treated as an access to IXh (the high byte of IX) Except if (IX+d) is used as well.
- Any access to L is treated as an access to IXl (the low byte of IX) Except if (IX+d) is used as well.

[1] Without the operand, that is.

- A DD prefix before a CB selects a completely different instruction set, see Section 3.5.

Some examples:

Without DD prefix	With DD prefix
`LD H,(HL)`	`LD H,(IX+d)`
`LD H,A`	`LD IXh,A`
`LD L,H`	`LD IXl,IXh`
`JP (HL)`	`JP (IX)`
`LD DE,0`	`LD DE,0`
`LD HL,0`	`LD IX,0`

3.3 FD Prefix [5]

This prefix has the same effect as the DD prefix, though IY is used instead of IX. Note `LD IXl,IYh` is not possible: only IX or IY is accessed in one instruction, never both.

3.4 ED Prefix [5]

There are a number of undocumented EDxx instructions, of which most are duplicates of documented instructions. Any instruction not listed has no effect (same behaviour as 2 `NOP` instructions).

The complete list except for the block instructions:

ED40	IN B,(C)	ED60	IN H,(C)
ED41	OUT (C),B	ED61	OUT (C),H
ED42	SBC HL,BC	ED62	SBC HL,HL
ED43	LD (nn),BC	ED63	LD (nn),HL
ED44	NEG	ED64	NEG**
ED45	RETN	ED65	RETN**
ED46	IM 0	ED66	IM 0**
ED47	LD I,A	ED67	RRD
ED48	IN C,(C)	ED68	IN L,(C)
ED49	OUT (C),C	ED69	OUT (C),L
ED4A	ADC HL,BC	ED6A	ADC HL,HL
ED4B	LD BC,(nn)	ED6B	LD HL,(nn)
ED4C	NEG**	ED6C	NEG**
ED4D	RETI	ED6D	RETN**
ED4E	IM 0**	ED6E	IM 0**
ED4F	LD R,A	ED6F	RLD
ED50	IN D,(C)	ED70	IN (C) / IN F,(C)**
ED51	OUT (C),D	ED71	OUT (C),0**
ED52	SBC HL,DE	ED72	SBC HL,SP
ED53	LD (nn),DE	ED73	LD (nn),SP
ED54	NEG**	ED74	NEG**
ED55	RETN**	ED75	RETN**
ED56	IM 1	ED76	IM 1**
ED57	LD A,I	ED77	NOP**
ED58	IN E,(C)	ED78	IN A,(C)

**Undocumented instruction

```
ED59  OUT (C),E        ED79  OUT (C),A
ED5A  ADC HL,DE        ED7A  ADC HL,SP
ED5B  LD DE,(nn)       ED7B  LD SP,(nn)
ED5C  NEG**            ED7C  NEG**
ED5D  RETN**           ED7D  RETN**
ED5E  IM 2             ED7E  IM 2**
ED5F  LD A,R           ED7F  NOP**
```

The ED70 instruction reads from I/O port C, but does not store the result. It just affects the flags like the other `IN x,(C)` instructions. ED71 simply outs the value 0 to I/O port C.

The ED63 is a duplicate of the 22 opcode (`LD (nn),HL`) and similarly ED6B is a duplicate of the 2A opcode. Of course the timings are different. These instructions are listed in the official documentation.

According to Gerton Lunter[2]:

> The instructions ED 4E and ED 6E are `IM 0` equivalents: when FF was put on the bus (physically) at interrupt time, the Spectrum continued to execute normally, whereas when an EF (`RST 28h`) was put on the bus it crashed, just as it does in that case when the Z80 is in the official interrupt mode 0. In `IM 1` the Z80 just executes a `RST 38h` (opcode FF) no matter what is on the bus.

All the `RETI/RETN` instructions are the same, all like the `RETN` instruction. So they all, including `RETI`, copy IFF2 to IFF1. More information on `RETI` and `RETN` and `IM x` is in section 5.3.

3.5 DDCB Prefix

The undocumented DDCB instructions store the result (if any) of the operation in one of the seven all-purpose registers, which one depends on the lower 3 bits of the last byte of the opcode (not operand, so not the offset).

000	B
001	C
010	D
011	E
100	H
101	L
110	(none: documented opcode)
111	A

The documented DDCB0106 is `RLC (IX+01h)`. So, clear the lower three bits (DDCB0100) and something is done to register B. The result of the RLC (which is stored in (IX+01h)) is now also stored in register B. Effectively, it does the following:

```
LD B,(IX+01h)
RLC B
LD (IX+01h),B
```

So you get double value for money. The result is stored in B and (IX+01h). The most common notation is: `RLC (IX+01h),B`

I've once seen this notation:

```
RLC (IX+01h)
LD B,(IX+01h)
```

[2] gerton@math.rug.nl

That's not correct: B contains the rotated value, even if (IX+01h) points to ROM. The DDCB SET and RES instructions do the same thing as the shift/rotate instructions:

```
DDCB10C0    SET 0,(IX+10h),B
DDCB10C1    SET 0,(IX+10h),C
DDCB10C2    SET 0,(IX+10h),D
DDCB10C3    SET 0,(IX+10h),E
DDCB10C4    SET 0,(IX+10h),H
DDCB10C5    SET 0,(IX+10h),L
DDCB10C6    SET 0,(IX+10h) - documented instruction
DDCB10C7    SET 0,(IX+10h),A
```

So for example with the last instruction, the value of (IX+10h) with bit 0 set is also stored in register A.

The DDCB BIT instructions do not store any value; they merely test a bit. That's why the undocumented DDCB BIT instructions are no different from the official ones:

```
DDCB d 78    BIT 7,(IX+d)
DDCB d 79    BIT 7,(IX+d)
DDCB d 7A    BIT 7,(IX+d)
DDCB d 7B    BIT 7,(IX+d)
DDCB d 7C    BIT 7,(IX+d)
DDCB d 7D    BIT 7,(IX+d)
DDCB d 7E    BIT 7,(IX+d) - documented instruction
DDCB d 7F    BIT 7,(IX+d)
```

3.6 FDCB Prefixes

Same as for the DDCB prefix, though IY is used instead of IX.

3.7 Combinations of Prefixes

This part may be of some interest to emulator coders. Here we define what happens if strange sequences of prefixes appear in the instruction cycle of the Z80.

If CB or ED is encountered, that byte plus the next make up an instruction. FD or DD should be seen as prefix setting a flag which says "use IX or IY in stead of HL", and not an instruction. In a large sequence of DD and FD bytes, it is the last one that counts. Also any other byte (or instruction) resets this flag.

```
FD DD 00 21 00 10    NOP NOP NOP LD HL,1000h
```

Chapter 4

Undocumented Effects

4.1 BIT instructions

`BIT n,r` behaves much like `AND r,`2^n with the result thrown away, and CF flag unaffected. Compare `BIT 7,A` with `AND 80h`: flag YF and XF are reset, SF is set if bit 7 was actually set; ZF is set if the result was 0 (bit was reset), and PF is effectively set if ZF is set (the result of the AND leaves either no bits set (PF set - parity even) or one bit set (PF reset - parity odd). So the rules for the flags are:

SF flag Set if n = 7 and tested bit is set.

ZF flag Set if the tested bit is reset.

YF flag Set if n = 5 and tested bit is set.

HF flag Always set.

XF flag Set if n = 3 and tested bit is set.

PF flag Set just like ZF flag.

NF flag Always reset.

CF flag Unchanged.

This is where things start to get strange. With the `BIT n,(IX+d)` instructions, the flags behave just like the `BIT n,r` instruction, except for YF and XF. These are not copied from the result but from something completely different, namely bit 5 and 3 of the high byte of IX+d (so IX plus the displacement).

Things get more bizarre with the `BIT n,(HL)` instruction. Again, except for YF and XF the flags are the same. YF and XF are copied from some sort of internal register. This register is related to 16 bit additions. Most instructions do not change this register. Unfortunately, I haven't tested all instructions yet, but here is the list so far.

`ADD HL,xx` Use the high byte of HL, ie. H before the addition.

`LD r,(IX+d)` Use high byte of the resulting address IX+d.

`JR d` Use high byte target address of the jump.

LD r,r' Doesn't change this register.

Any help here would be most appreciated!

4.2 Memory Block Instructions [1]

The `LDI/LDIR/LDD/LDDR` instructions affect the flags in a strange way. At every iteration, a byte is copied. Take that byte and add the value of register A to it. Call that value n. Now, the flags are:

YF flag A copy of bit 1 of n.

HF flag Always reset.

XF flag A copy of bit 3 of n.

PF flag Set if BC not 0.

SF, ZF, CF flags These flags are unchanged.

And now for `CPI/CPIR/CPD/CPDR`. This instruction compares a series of bytes in memory to register A. Effectively, it can be said it does `CP (HL)` at every iteration. The result of that compare sets the HF flag, which is important for the next step. Take the value of register A, subtract the value of the memory address, and finally subtract the value of HF flag, which is set or reset by the hypothetical `CP (HL)`. So, `n = A - (HL) - HF`.

SF, ZF, HF flags Set by the hypothetical `CP (HL)`.

YF flag A copy of bit 1 of n.

XF flag A copy of bit 3 of n.

PF flag Set if BC is not 0.

NF flag Always set.

CF flag Unchanged.

4.3 I/O Block Instructions

These are the most be bizarre instructions, as far as flags is concerned. Ramsoft found all of the flags. The out instructions behave differently than the in instructions, which doesn't make the CPU very symmetrical.

First of all, all instructions affect the following flags:

SF, ZF, YF, XF flags Affected by decreasing register B, as in `DEC B`.

NF flag A copy of bit 7 of the value read from or written to an I/O port.

And now the for `OUTI/OTIR/OUTD/OTDR` instructions. Take state of the L after the increment or decrement of HL; add the value written to the I/O port to; call that k for now. If $k > 255$, then the CF and HF flags are set. The PF flags is set like the parity of k bitwise and'ed with 7, bitwise xor'ed with B.

HF and CF Both set if $((HL) + L > 255)$

PF The parity of $((((HL) + L) \& 7) \text{ xor } B)$

`INI/INIR/IND/INDR` use the C register in stead of the L register. There is a catch though, because not the value of C is used, but C + 1 if it's `INI/INIR` or C - 1 if it's `IND/INDR`. So, first of all `INI/INIR`:

HF and CF Both set if $((HL) + ((C + 1) \& 255) > 255)$

PF The parity of (((HL) + ((C + 1) & 255)) & 7) xor B)

And last `IND/INDR`:

HF and CF Both set if ((HL) + ((C - 1) & 255) > 255)

PF The parity of (((HL) + ((C - 1) & 255)) & 7) xor B)

4.4 16 Bit I/O ports

Officially the Z80 has an 8 bit I/O port address space. When using the I/O ports, the 16 address lines are used. And in fact, the high 8 bit do actually have some value, so you can use 65536 ports after all. `IN r,(C)`, `OUT (C),r`, and the Block I/O instructions actually place the entire BC register on the address bus. Similarly `IN A,(n)` and `OUT (n),A` put A $\times$ 256 + n on the address bus.

The `INI/INIR/IND/INDR` instructions use BC after decrementing B, and the `OUTI/OTIR/OUTD/OTDR` instructions before.

4.5 Block Instructions

The repeated block instructions simply decrease the PC by two so the instruction is simply re-executed. So interrupts can occur during block instructions. So, `LDIR` is simply `LDI` + if BC is not 0, decrease PC by 2.

4.6 16 Bit Additions

The 16 bit additions are a bit more complicated than 8 bit ones. Since the Z80 is an 8-bit CPU, 16 bit additions are done in two stages: first the lower bytes are added, then the two higher bytes. The SF, YF, HF, XF flags are affected as by the second (high) 8 bit addition. ZF is set if the whole 16 bit result is 0.

4.7 DAA Instruction

This instruction is useful when you're using BCD values. After an addition or subtraction, `DAA` corrects the value back to BCD again. Note that it uses the CF flag, so it cannot be used after `INC` and `DEC`.

Stefano Donati from Ramsoft[1] has found the tables which describe the `DAA` operation. The input is the A register and the CF, NF, HF flags. Result is as follows:

Depending on the NF flag, the 'diff' from this table must be added (NF is reset) or subtracted (NF is set) to A.

[1] http://www.ramsoft.bbk.org/

CF	high nibble	HF	low nibble	diff
0	0-9	0	0-9	00
0	0-9	1	0-9	06
0	0-8	*	a-f	06
0	a-f	0	0-9	60
1	*	0	0-9	60
1	*	1	0-9	66
1	*	*	a-f	66
0	9-f	*	a-f	66
0	a-f	1	0-9	66

The CF flag is affected as follows:

CF	high nibble	low nibble	CF'
0	0-9	0-9	0
0	0-8	a-f	0
0	9-f	a-f	1
0	a-f	0-9	1
1	*	*	1

The HF flags is affected as follows:

NF	HF	low nibble	HF'
0	*	0-9	0
0	*	a-f	1
1	0	*	0
1	1	6-f	0
1	1	0-5	1

SF, YF, XF are copies of bit 7,5,3 of the result respectively; ZF is set according to the result and NF is always unchanged.

Chapter 5

Interrupts

There are two types of interrupts, maskable and non-maskable. The maskable type is ignored if IFF1 is reset. Non-maskable interrupts (NMI) will are always accepted, and they have a higher priority, so if the two are requested at the same time the NMI will be accepted first.

For the interrupts, the following things are important: Interrupt Mode (set with the `IM 0, IM 1, IM 2` instructions), the interrupt flip-flops (IFF1 and IFF2), and the I register. When a maskable interrupt is accepted, a external device can put a value on the databus.

Both types of interrupts increase the R register by one, when accepted.

5.1 Non-Maskable Interrupts (NMI)

When a NMI is accepted, IFF1 is reset. At the end of the routine, IFF1 must be restored (so the running program is not affected). That's why IFF2 is there; to keep a copy of IFF1.

An NMI is accepted when the NMI pin on the Z80 is made low (edge-triggered). The Z80 responds to the *change* of the line from +5 to 0 — so the interrupt line doesn't have a state, it's just a pulse. When this happens, a call is done to address 0066h and IFF1 is reset so the routine isn't bothered by maskable interrupts. The routine should end with an `RETN` (RETurn from Nmi) which is just a usual RET, but also copies IFF2 to IFF1, so the IFFs are the same as before the interrupt.

You can check whether interrupts were disabled or not during an NMI by using the `LD A,I` or `LD A,R` instruction. These instructions copy IFF2 to the PF flag.

Accepting an `NMI` costs 11 t-states.

5.2 Maskable Interrupts (INT)

If the INT line is low and IFF1 is set, a maskable interrupt is accepted — whether or not the last INT routine has finished. That's why you should not enable interrupts during such a routine, and make sure that the device that generated it has put the INT line up again before ending the routine. So unlike NMI interrupts, the interrupt line has a state; it's not a pulse.

When an INT is accepted, both IFF1 and IFF2 are cleared, preventing another interrupt from occurring which would end up as an infinite loop (and overflowing the stack). What happens next depends on the Interrupt Mode.

A device can place a value on the databus when the interrupt is accepted. Some computer systems do not utilize this feature, and this value ends up being FFh.

Interrupt Mode 0 This is the 8080 compatibility mode. The instruction on the bus is executed (usually an `RST` instruction, but it can be anything. The I register is not used. Assuming it a `RST` instruction, accepting this takes 13 t-states.

Interrupt Mode 1 An RST 38h is executed, no matter what value is put on the bus or what value the I register has. Accepting this type costs 13 t-states.

Interrupt Mode 2 A call is made to the address read from memory. What address is read from is calculated as follows: (I register) × 256 + (value on bus). Zilog's user manual states (very convincingly) that the least significant bit of the address is always 0, so they calculate the address that is read from as: (I register) × 256 + (value on bus & 0xFE). I have tested this and it not correct. Of course a word (two bytes) are read, making the address where the call is made to. In this way, you can have a vector table for interrupts. Accepting this of interrupt type costs 19 t-states.

At the end of a maskable interrupt, the interrupts should be enabled again. You can assume that was the state of the IFFs because otherwise the interrupt wasn't accepted. So, an INT routine always ends with an `EI` and a `RET` (`RETI` according to the official documentation, more about that later):

```
INT:
    .
    .
    .
    EI
    RETI or RET
```

Note a fact about `EI`: a maskable interrupt isn't accepted directly after it, so the next opportunity for an interrupt is after the `RETI`. This is very useful; if the INT line is still low, an interrupt is accepted again. If this happens a lot and the interrupt is generated before the `RETI`, the stack could overflow (since the routine would be called again and again). But this property of `EI` prevents this.

`DI` is not necessary at the start of the interrupt routine: the interrupt flip-flops are cleared when accepting the interrupt.

You can use `RET` instead of `RETI`, depending on the hardware setup. `RETI` is only useful if you have something like a Z80 PIO to support daisy-chaining: queuing interrupts. The PIO can detect that the routine has ended by the opcode of `RETI`, and let another device generate an interrupt. That is why I called all the undocumented EDxx RET instructions `RETN`: All of them operate alike, the only difference of `RETI` is its specific opcode which the Z80 PIO recognises.

5.3 Things affecting the Interrupt flip-flops

All the IFF related things are:

	IFF1	IFF2	
CPU reset	0	0	
`DI`	0	0	
`EI`	1	1	
Accept INT	0	0	
Accept NMI	0	-	
`RETI/N`	IFF2	-	All the EDxx RETI/N instructions
`LD A,I/LD A,R`	-	-	Copies IFF2 into PF flag

If you're working with a Z80 system without NMIs (like the MSX), you can forget all about the two separate IFFs; since a NMI isn't ever generated, the two will always be the same.

Some documentation says that when an NMI is accepted, IFF1 is first copied into IFF2 before IFF1 is cleared. If this is true, the state of IFF2 is lost after a nested NMI, which is undesirable. Have tested this in the following way: make sure the Z80 is in `EI` mode, generate an NMI. In the

NMI routine, wait for another NMI before executing `RETN`. In the second NMI IFF2 was still set, so IFF1 is *not* copied to IFF2 when accepting an NMI.

Another interesting fact is this. I was trying to figure out whether the undocumented ED `RET` instructions were `RETN` or `RETI`. I tested this by putting the machine in `EI` mode, wait for an NMI and end with one of the ED `RET` instructions. Then execute a `HALT` instruction. If IFF1 was not restored, the machine would hang but this did not happen with any of the instructions, including the documented `RETI`!

Since every INT routine must end with `EI` followed by `RETI` officially, It does not matter that RETI copies IFF2 into IFF1; both are set anyway.

5.4 HALT instruction

The HALT instruction halts the Z80; it does not increase the PC so that the instruction is re-executed, until a maskable or non-maskable interrupt is accepted. Only then does the Z80 increase the PC again and continues with the next instruction. During the HALT state, the HALT line is set. The PC is increased before the interrupt routine is called.

5.5 Where interrupts are accepted

During execution of instructions, interrupts won't be accepted. Only *between* instructions. This is also true for prefixed instructions.

Directly after an EI or DI instruction, interrupts aren't accepted. They're accepted again after the instruction after the EI (RET in the following example). So for example, look at this MSX2 routine that reads a scanline from the keyboard:

```
LD    C,A
DI
IN    A,(0AAh)
AND   0F0h
ADD   A,C
OUT   (0AAh),A
EI
IN    A,(0A9h)
RET
```

You can assume that there never is an interrupt after the `EI`, before the `IN A,(0A9h)` — which would be a problem because the MSX interrupt routine reads the keyboard too.

Using this feature of `EI`, it is possible to check whether it is true that interrupts are never accepted during instructions:

```
    DI
    make sure INT is active
    EI
    insert instruction to test
INT:
    store PC where INT was accepted
    RET
```

And yes, for all instructions, including the prefixed ones, interrupts are never accepted during an instruction. Only after the tested instruction. Remember that block instructions simply re-execute themselves (by decreasing the PC with 2) so an interrupt is accepted after each iteration.

Another predictable test is this: at the "insert instruction to test" insert a large sequence of `EI` instructions. Of course, during execution of the `EI` instructions, no interrupts are accepted.

But now for the interesting stuff. ED or CB make up instructions, so interrupts are accepted after them. But DD and FD are prefixes, which only slightly affects the next opcode. If you test a large sequence of DDs or FDs, the same happens as with the `EI` instruction: no interrupts are accepted during the execution of these sequences.

This makes sense, if you think of DD and FD as a prefix which set the "use IX instead of HL" or "use IY instead of HL" flag. If an interrupt was accepted after DD or FD, this flag information would be lost, and:

```
DD 21 00 00    LD IX,0
```

could be interpreted as a simple `LD HL,0` if the interrupt was after the last DD. Which never happens, so the implementation is correct. Although I haven't tested this, as I imagine the same holds for NMI interrupts.

Chapter 6

Timing and R register

6.1 R register and memory refresh

During every first machine cycle (beginning of an instruction or part of it — prefixes have their own M1 two), the memory refresh cycle is issued. The whole IR register is put on the address bus, and the RFSH pin is lowered. It is unclear whether the Z80 increases the R register before or after putting IR on the bus.

The R register is increased at every first machine cycle (M1). Bit 7 of the register is never changed by this; only the lower 7 bits are included in the addition. So bit 7 stays the same, but it can be changed using the `LD R,A` instruction.

Instructions without a prefix increase R by one. Instructions with an ED, CB, DD, FD prefix, increase R by two, and so do the DDCBxxxx and FDCBxxxx instructions (weird enough). Just a stray DD or FD increases the R by one. `LD A,R` and `LD R,A` access the R register after it is increased (by the instruction itself).

Remember that the block instructions simply decrease the PC with two, so the instructions are re-executed. So LDIR increased R by BC times 2 (note that in the case of BC = 0, R is increased by 10000h times 2, effectively 0).

Accepting an maskable or non-maskable interrupt increases the R by one.

After a hardware reset, or after power on, the R register is reset to 0.

That should cover all there is to say about the R register. It is often used in programs for a random value, which is good but of course not truly random.

Chapter 7

Other Information

7.1 Errors in official documentation

In some official Zilog documentation, the are some errors. Some don't have all of these mistakes, so your documentation may not be flawed but these are just things to look out for.

- The Flag affection summary table shows that `LDI/LDIR/LDD/LDDR` instructions leave the SF and ZF in an undefined state. This is not correct; the SF and ZF flags are unaffected (like the same documentation says).
- Similarly, the same table shows that `CPI/CPIR/CPD/CPDR` leave the SF and HF flags in an undefined state. Not true, they are affected as defined elsewhere in the documentation.
- Also, the table says about `INI/OUTD`/etc "Z=0 if B <> 0 otherwise Z=0"; of course the latter should be Z=1.
- The `INI/INIR/IND/INDR/OUTI/OUTD/OTIR/OTDR` instructions do affect the CF flag (some official documentation says they leave it unaffected, important!) and the NF flag isn't always set but may also be reset (see 4.3 for exact operation).
- When an NMI is accepted, the IFF1 isn't copied to IFF2. Only IFF1 is reset.
- In the 8-bit Load Group, the last two bits of the second byte of the `LD r,(IX + d)` opcode should be 10 and not 01.
- In the 16-bit Arithmetic Group, bit 6 of the second byte of the `ADD IX, pp` opcode should be 0, not 1.
- `IN x,(C)` resets the HF flag, it never sets it. Some documentation states it is set according to the result of the operation; this is impossible since no arithmetic is done in this instruction.

Note: In zilog's own Z80 User Manual (z80cpu_um.pdf), there are also errors, some are very confusing, I will mention the ones I have found here:

- page 21, figure 2 says the Alternative Register Set contains 2 B' registers, this should ofcourse be B' and C'.
- page 26, figure 16 shows very convincingly that the least significant bit of the address to read for Interrupt Mode 2 is always 0. I have tested this and it is not correct, it can also be 1, in my testcase the bus contained 0xFF and the address that was read did not end in 0xFE but was 0xFF.

Bibliography

[1] Mark Rison Z80 page for !CPC.
`http://www.acorn.co.uk/~mrison/en/cpc/tech.html`

[2] YAZE (Yet Another Z80 Emulator). This is a CPM emulator by Frank Cringle. It emulates almost every undocumented flag, very good emulator. Also includes a very good instruction exerciser and is released under the GPL.
`ftp://ftp.ping.de/pub/misc/emulators/yaze-1.10.tar.gz`
Note: the instruction exerciser zexdoc/zexall does not test I/O instructions and not all normal instructions (for instance LD A,(IX+n) is tested, but not with different values of n, just n=1, values above 128 (LD A,(IX-n) are not tested) but it still gives a pretty good idea of how well a simulated Z80 works.

[3] Z80 Family Official Support Page by Thomas Scherrer. Very good – your one-stop Z80 page.
`http://www.geocities.com/SiliconValley/Peaks/3938/z80_home.htm`

[4] Spectrum FAQ technical information.
`http://www.worldofspectrum.org/faq/`

[5] Gerton Lunter's Spectrum emulator (Z80). In the package there is a file TECHINFO.DOC, which contains a lot of interesting information. Note that the current version can only be unpacked in Windows.
`ftp://ftp.void.jump.org/pub/sinclair/emulators/pc/dos/z80-400.zip`

[6] Mostek Z80 Programming Manual – a very good reference to the Z80.

[7] Z80 Product Specification, from MSX2 Hardware Information.
`http://www.hardwareinfo.msx2.com/pdf/Zilog/z80.pdf`

Chapter 8

Instruction Tables

8.1 8-Bit Load Group

	Symbolic				Flags					Opcode			M	T		
Mnemonic	Operation	SF	ZF	YF	HF	XF	PF	NF	CF	76 543 210	Hex	Bytes	Cycles	States	Comments	
LD r,r'	r←r'	●	●	●	●	●	●	●	●	01 r r'		1	1	4	r,r'	Reg
LD p,p'	p←p'	●	●	●	●	●	●	●	●	11 011 101	DD	2	2	8	000	B
										01 p p'					001	C
LD q,q'	q←q'	●	●	●	●	●	●	●	●	11 111 101	FD	2	2	8	010	D
										01 q q'					011	E
LD r,n	r←n	●	●	●	●	●	●	●	●	00 r 110		2	2	7	100	H
										← n →					101	L
LD p,n	p←n	●	●	●	●	●	●	●	●	11 011 101	DD	3	3	11	111	A
										00 p 110						
										← n →						
LD q,n	q←n	●	●	●	●	●	●	●	●	11 111 101	FD	3	3	11	p,p'	Reg
										00 q 110					000	B
										← n →					001	C
LD r,(HL)	r←(HL)	●	●	●	●	●	●	●	●	01 r 110		1	2	7	010	D
LD r,(IX+d)	r←(IX+d)	●	●	●	●	●	●	●	●	11 011 101	DD	3	5	19	011	E
										01 r 110					100	IXh
										← d →					101	IXl
LD r,(IY+d)	r←(IY+d)	●	●	●	●	●	●	●	●	11 111 101	FD	3	5	19	111	A
										01 r 110						
										← d →						
LD (HL),r	(HL)←r	●	●	●	●	●	●	●	●	01 110 r		1	2	7	q,q'	Reg
LD (IX+d),r	(IX+d)←r	●	●	●	●	●	●	●	●	11 011 101	DD	3	5	19	000	B
										01 110 r					001	C
										← d →					010	D
LD (IY+d),r	(IY+d)←r	●	●	●	●	●	●	●	●	11 111 101	FD	3	5	19	011	E
										01 110 r					100	IYh
										← d →					101	IYl
LD (HL),n	(HL)←n	●	●	●	●	●	●	●	●	00 110 110	36	2	3	10	111	A
										← n →						
LD (IX+d),n	(IX+d)←n	●	●	●	●	●	●	●	●	11 011 101	DD	4	5	19		
										00 110 110	36					
										← d →						
										← n →						
LD (IY+d),n	(IY+d)←n	●	●	●	●	●	●	●	●	11 111 101	FD	4	5	19		
										00 110 110	36					
										← d →						
										← n →						
LD A,(BC)	A←(BC)	●	●	●	●	●	●	●	●	00 001 010	0A	1	2	7		
LD A,(DE)	A←(DE)	●	●	●	●	●	●	●	●	00 011 010	1A	1	2	7		
LD A,(nn)	A←(nn)	●	●	●	●	●	●	●	●	00 111 010	3A	3	4	13		
										← n →						
										← n →						

(continued)

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
LD (BC),A	(BC)←A	•	•	•	•	•	•	•	•	00 000 010	02	1	2	7	
LD (DE),A	(DE)←A	•	•	•	•	•	•	•	•	00 010 010	12	1	2	7	
LD (nn),A	(nn)←A	•	•	•	•	•	•	•	•	00 110 010	32	3	4	13	
										← n →					
										← n →					
LD A,I	A←I	↕	↕	↕	0	↕	IFF2	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A,R	A←R	↕	↕	↕	0	↕	IFF2	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I,A	I←A	•	•	•	•	•	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R,A	R←A	•	•	•	•	•	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

8.2 16-Bit Load Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
LD dd,nn	dd←nn	•	•	•	•	•	•	•	•	00 dd0 001		3	3	10	dd Reg
										← n →					00 BC
										← n →					01 DE
LD IX,nn	IX←nn	•	•	•	•	•	•	•	•	11 011 101	DD	4	4	14	10 HL
										00 100 001	21				11 SP
										← n →					
										← n →					
LD IY,nn	IX←nn	•	•	•	•	•	•	•	•	11 111 101	FD	4	4	14	
										00 100 001	21				
										← n →					
										← n →					
LD HL,(nn)	H←(nn+1)	•	•	•	•	•	•	•	•	00 101 010	2A	3	5	16	
	L←(nn)									← n →					
										← n →					
LD dd,(nn)	ddh←(nn+1)	•	•	•	•	•	•	•	•	11 101 101	ED	4	6	20	
	ddl←(nn)									01 dd1 011					
										← n →					
										← n →					
LD IX,(nn)	IXh←(nn+1)	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	20	
	IXl←(nn)									00 101 010	2A				
										← n →					
										← n →					
LD IY,(nn)	IYh←(nn+1)	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	20	
	IYl←(nn)									00 101 010	2A				
										← n →					
										← n →					
LD (nn),HL	(nn+1)←H	•	•	•	•	•	•	•	•	00 100 010	22	3	5	16	
	(nn)←L									← n →					
										← n →					
LD (nn),dd	(nn+1)←ddh	•	•	•	•	•	•	•	•	11 101 101	ED	4	6	20	
	(nn)←ddl									01 dd0 011					
										← n →					
										← n →					
LD (nn),IX	(nn+1)←IXh	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	20	
	(nn)←IXl									00 100 010	22				
										← n →					
										← n →					
LD (nn),IY	(nn+1)←IYh	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	20	
	(nn)←IYl									00 100 010	22				
										← n →					
										← n →					
LD SP,HL	SP←HL	•	•	•	•	•	•	•	•	11 111 001	F9	1	1	6	
LD SP,IX	SP←IX	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	
										11 111 001	F9				
LD SP,IY	SP←IY	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	
										11 111 001	F9				

(continued)

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
PUSH qq	(SP-2)←qql	•	•	•	•	•	•	•	•	11 qq0 101		1	3	11	qq Reg
	(SP-1)←qqh														00 BC
	SP←SP-2														01 DE
PUSH IX	(SP-2)←IXl	•	•	•	•	•	•	•	•	11 011 101	DD	2	4	15	10 HL
	(SP-1)←IXh									11 100 101	E5				11 AF
	SP←SP-2														
PUSH IY	(SP-2)←IYl	•	•	•	•	•	•	•	•	11 111 101	FD	2	4	15	
	(SP-1)←IYh									11 100 101	E5				
	SP←SP-2														
POP qq	qqh←(SP+1)	•	•	•	•	•	•	•	•	11 qq0 001		1	3	10	
	qql←(SP)														
	SP←SP+2														
POP IX	IXh←(SP+1)	•	•	•	•	•	•	•	•	11 011 101	DD	2	4	14	
	IXl←(SP)									11 100 001	E1				
	SP←SP+2														
POP IY	IYh←(SP+1)	•	•	•	•	•	•	•	•	11 111 101	FD	2	4	14	
	IYl←(SP)									11 100 001	E1				
	SP←SP+2														

8.3 Exchange, Block Transfer, Search Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
EX DE,HL	DE↔HL	•	•	•	•	•	•	•	•	11 101 011	EB	1	1	4	
EX AF,AF'	AF↔AF'	•	•	•	•	•	•	•	•	00 001 000	08	1	1	4	
EXX	BC↔BC'	•	•	•	•	•	•	•	•	11 011 001	D9	1	1	4	
	DE↔DE'														
	HL↔HL'														
EX (SP),HL	H↔(SP+1)	•	•	•	•	•	•	•	•	11 100 011	E3	1	5	19	
	L↔(SP)														
EX (SP),IX	IXh↔(SP+1)	•	•	•	•	•	•	•	•	11 011 101	DD	2	6	23	
	IXl↔(SP)									11 100 011					
EX (SP),IY	IYh↔(SP+1)	•	•	•	•	•	•	•	•	11 111 101	FD	2	6	23	
	IYl↔(SP)									11 100 011					
LDI	(DE)←(HL)	•	•	↕[4]	0	↕[4]	↕[1]	0	•	11 101 101	ED	2	4	16	
	DE←DE+1									10 100 000	A0				
	HL←HL+1														
	BC←BC-1														
LDIR	(DE)←(HL)	•	•	↕[4]	0	↕[4]	0[2]	0	•	11 101 101	ED	2	5	21	if BC≠0
	DE←DE+1									10 110 000	B0	2	4	16	if BC=0
	HL←HL+1														
	BC←BC-1														
	Repeat until BC=0														
LDD	(DE)←(HL)	•	•	↕[4]	0	↕[4]	↕[1]	0	•	11 101 101	ED	2	4	16	
	DE←DE-1									10 101 000	A8				
	HL←HL-1														
	BC←BC-1														
LDDR	(DE)←(HL)	•	•	↕[4]	0	↕[4]	0[2]	0	•	11 101 101	ED	2	5	21	if BC≠0
	DE←DE-1									10 111 000	B8	2	4	16	if BC=0
	HL←HL-1														
	BC←BC-1														
	Repeat until BC=0														
CPI	A-(HL)	↕[4]	↕[3]	↕[4]	↕[4]	↕[4]	↕[1]	1	•	11 101 101	ED	2	4	16	
	HL←HL+1									10 100 001	A1				
	BC←BC-1														
CPIR	A-(HL)	↕[4]	↕[3]	↕[4]	↕[4]	↕[4]	↕[1]	1	•	11 101 101	ED	2	5	21	if BC≠0 and A≠(HL)
	HL←HL+1									10 110 001	B1	2	4	16	if BC=0 or A=(HL)
	BC←BC-1														
	Repeat until A=(HL) or BC=0														
CPD	A-(HL)	↕[4]	↕[3]	↕[4]	↕[4]	↕[4]	↕[1]	1	•	11 101 101	ED	2	4	16	
	HL←HL-1									10 101 001	A9				
	BC←BC-1														
CPDR	A-(HL)	↕[4]	↕[3]	↕[4]	↕[4]	↕[4]	↕[1]	1	•	11 101 101	ED	2	5	21	if BC≠0 and A≠(HL)
	HL←HL-1									10 111 001	B9	2	4	16	if BC=0 or A=(HL)
	BC←BC-1														
	Repeat until A=(HL) or BC=0														

Note:
[1]PF is 0 the result of BC-1=0, otherwise PF is set.
[2]PF is 0 only at completion of the instruction.
[3]ZF is set if A=(HL), otherwise ZF is reset.
[4]See section 4.2 for a description.

8.4 8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	76	543	210	Hex	Bytes	M Cycles	T States	Comments
ADD A,r	A←A+r	↕	↕	↕	↕	↕	VF	0	↕	10	000	r		1	1	4	r Reg
ADD A,p	A←A+p	↕	↕	↕	↕	↕	VF	0	↕	11	011	101	DD	2	2	8	000 B
										10	000	p					001 C
ADD A,q	A←A+q	↕	↕	↕	↕	↕	VF	0	↕	11	111	101	FD	2	2	8	010 D
										10	000	q					011 E
ADD A,n	A←A+n	↕	↕	↕	↕	↕	VF	0	↕	11	000	110		2	2	7	100 H
											← n →						101 L
ADD A,(HL)	A←A+(HL)	↕	↕	↕	↕	↕	VF	0	↕	10	000	110		1	2	7	111 A
ADD A,(IX+d)	A←A+(IX+d)	↕	↕	↕	↕	↕	VF	0	↕	11	011	101	DD	3	5	19	
										10	000	110					
											← d →						p Reg
ADD A,(IY+d)	A←A+(IY+d)	↕	↕	↕	↕	↕	VF	0	↕	11	111	101	FD	3	5	19	000 B
										10	000	110					001 C
											← d →						010 D
ADC A,s	A←A+s+CF	↕	↕	↕	↕	↕	VF	0	↕		001						011 E
SUB s	A←A-s	↕	↕	↕	↕	↕	VF	1	↕		010						100 IXh
SBC A,s	A←A-s-CF	↕	↕	↕	↕	↕	VF	1	↕		011						101 IXl
AND s	A←A∧s	↕	↕	↕	1	↕	PF	0	0		100						111 A
OR s	A←A∨s	↕	↕	↕	0	↕	PF	0	0		110						
XOR s	A←A•s	↕	↕	↕	0	↕	PF	0	0		101						
CP s	A-s	↕	↕	↕[1]	↕	↕[1]	VF	1	↕		111						q Reg
INC r	r←r+1	↕	↕	↕	↕	↕	VF	0	•	00	r	100		1	1	4	000 B
INC p	p←p+1	↕	↕	↕	↕	↕	VF	0	•	11	011	101	DD	2	2	8	001 C
										00	p	100					010 D
INC q	q←q+1	↕	↕	↕	↕	↕	VF	0	•	11	111	101	FD	2	2	8	011 E
										00	q	100					100 IYh
INC (HL)	(HL)←(HL)+1	↕	↕	↕	↕	↕	VF	0	•	00	110	100		1	3	11	101 IYl
INC (IX+d)	(IX+d)←(IX+d)+1	↕	↕	↕	↕	↕	VF	0	•	11	011	101	DD	3	6	23	111 A
										00	110	100					
											← d →						
INC (IY+d)	(IY+d)←(IY+d)+1	↕	↕	↕	↕	↕	VF	0	•	11	111	101	FD	3	6	23	
										00	110	100					
											← d →						
DEC m	m←m-1	↕	↕	↕	↕	↕	VF	1	•			101					

Note: [1]YF and XF flags are copied from the operand s, not the result A-s
s is any of r, p, q, n, (HL), (IX+d), (IY+d) as shown for ADD. The indicated bits replace the 000 in the ADD set above
m is any of r, p, q, (HL), (IX+d), (IY+d) as shown for INC. Replace 100 with 101 in opcode

8.5 General-Purpose Arithmetic and CPU Control Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	76	543	210	Hex	Bytes	M Cycles	T States	Comments
DAA		↕	↕	↕	↕	↕	PF	•	↕	00	100	111	27	1	1	4	Decimal adjust accumulator
CPL	A←$\overline{A}$	•	•	↕	1	↕	•	1	•	00	101	111	2F	1	1	4	Compliment
NEG	A←0-A	↕	↕	↕	↕	↕	VF	1	↕	11	101	101	ED	2	2	8	Negate
										01	000	100	44				
CCF	CF←$\overline{CF}$	•	•	↕[1]	↕[2]	↕[1]	•	0	↕	00	111	111	3F	1	1	4	
SCF	CF←1	•	•	↕[1]	0	↕[1]	•	0	1	00	110	111	37	1	1	4	
NOP		•	•	•	•	•	•	•	•	00	000	000	00	1	1	4	
HALT		•	•	•	•	•	•	•	•	01	110	110	76	1	1	4	
DI[3]	IFF1,2←0	•	•	•	•	•	•	•	•	11	110	011	F3	1	1	4	
EI[3]	IFF1,2←1	•	•	•	•	•	•	•	•	11	111	011	FB	1	1	4	
IM 0[4]		•	•	•	•	•	•	•	•	11	101	101	ED	2	2	8	
										01	000	110	46				
IM 1[4]		•	•	•	•	•	•	•	•	11	101	101	ED	2	2	8	
										01	010	110	56				
IM 2[4]		•	•	•	•	•	•	•	•	11	101	101	ED	2	2	8	
										01	011	110	5E				

Note: [1]YF and XF are copied from register A.
[2]HF is like CF before the instruction.
[3]No interrupts are accepted directly after EI or DI.
[4]This instruction has other undocumented opcodes.

8.6 16-Bit Arithmetic Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
ADD HL,ss	HL←HL+ss	•	•	$\updownarrow^{2}$	$\updownarrow^{2}$	$\updownarrow^{2}$	•	0	$\updownarrow^{1}$	00 ss1 001		1	3	11	ss Reg
ADC HL,ss	HL←HL+ss+CF	$\updownarrow^{1}$	$\updownarrow^{1}$	$\updownarrow^{2}$	$\updownarrow^{2}$	$\updownarrow^{2}$	VF^{1}	0	$\updownarrow^{1}$	11 101 101	ED	2	4	15	00 BC
										01 ss1 010					01 DE
SBC HL,ss	HL←HL-ss-CF	$\updownarrow^{1}$	$\updownarrow^{1}$	$\updownarrow^{2}$	$\updownarrow^{2}$	$\updownarrow^{2}$	VF^{1}	1	$\updownarrow^{1}$	11 101 101	ED	2	4	15	10 HL
										01 ss0 010					11 SP
ADD IX,pp	IX←IX+pp	•	•	$\updownarrow^{2}$	$\updownarrow^{2}$	$\updownarrow^{2}$	•	0	$\updownarrow^{1}$	11 011 110	DD	2	4	15	
										00 pp1 001					pp Reg
ADD IY,qq	IY←IY+qq	•	•	$\updownarrow^{2}$	$\updownarrow^{2}$	$\updownarrow^{2}$	•	0	$\updownarrow^{1}$	11 111 110	FD	2	4	15	00 BC
										00 qq1 001					01 DE
INC ss	ss←ss+1	•	•	•	•	•	•	•	•	00 ss0 011		1	1	6	10 IX
INC IX	IX←IX+1	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	11 SP
										00 100 011	23				
INC IY	IY←IY+1	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	qq Reg
										00 100 011	23				00 BC
DEC ss	ss←ss-1	•	•	•	•	•	•	•	•	00 ss1 011		1	1	6	01 DE
DEC IX	IX←IX-1	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	10 IY
										00 101 011	2B				11 SP
DEC IY	IY←IY-1	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	
										00 101 011	2B				

Note: [1]Flag is affected by the 16 bit result.
[2]Flag is affected by the high-byte addition.

8.7 Rotate and Shift Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76	543	210	Hex	Bytes	M Cycles	T States	Comments
RLCA	[CF]←[7←0]←	•	•	↕	0	↕	•	0	↕	00	000	111	07	1	1	4	
RLA	[CF]←[7←0]←	•	•	↕	0	↕	•	0	↕	00	010	111	17	1	1	4	
RRCA	→[7←0]→[CF]	•	•	↕	0	↕	•	0	↕	00	001	111	0F	1	1	4	
RRA	→[7←0]→[CF]	•	•	↕	0	↕	•	0	↕	00	011	111	1F	1	1	4	
RLC r	[CF]←[7←0]←	↕	↕	↕	0	↕	PF	0	↕	11 00	001 [000]	011 r	CB	2	2	8	r Reg 000 B
RLC (HL)	[CF]←[7←0]←	↕	↕	↕	0	↕	PF	0	↕	11 00	001 [000]	011 110	CB	2	4	15	001 C 010 D
RLC (IX+d)	[CF]←[7←0]←	↕	↕	↕	0	↕	PF	0	↕	11 11 ← 00	011 001 d [000]	101 011 → 110	DD CB	4	6	23	011 E 100 H 101 L 111 A
RLC (IY+d)	[CF]←[7←0]←	↕	↕	↕	0	↕	PF	0	↕	11 11 ← 00	111 001 d [000]	101 011 → 110	FD CB	4	6	23	
RLC (IX+d),r	r←(IX+d) RLC r (IX+d)←r	↕	↕	↕	0	↕	PF	0	↕	11 11 ← 00	011 001 d [000]	101 011 → r	DD CB	4	6	23	
RLC (IY+d),r	r←(IY+d) RLC r (IY+d)←r	↕	↕	↕	0	↕	PF	0	↕	11 11 ← 00	111 001 d [000]	101 011 → r	FD CB	4	6	23	
RL m	[CF]←[7←0]←	↕	↕	↕	0	↕	PF	0	↕		[010]						
RRC m	→[7←0]→[CF]	↕	↕	↕	0	↕	PF	0	↕		[001]						
RR m	→[7←0]→[CF]	↕	↕	↕	0	↕	PF	0	↕		[011]						
SLA m	[CF]←[7←0]←0	↕	↕	↕	0	↕	PF	0	↕		[100]						
SLL m	[CF]←[7←0]←1	↕	↕	↕	0	↕	PF	0	↕		[110]						
SRA m	→[7→0]→[CF]	↕	↕	↕	0	↕	PF	0	↕		[101]						
SRL m	0→[7→0]→[CF]	↕	↕	↕	0	↕	PF	0	↕		[111]						
RLD	A [7-4|3-0] [7-4|3-0] (HL)	↕	↕	↕	0	↕	PF	0	•	11 01	101 101	101 111	ED 6F	2	5	18	
RRD	A [7-4|3-0] [7-4|3-0] (HL)	↕	↕	↕	0	↕	PF	0	•	11 01	101 100	101 111	ED 67	2	5	18	

Note: m is one of r,(HL),(IX+d),(IY+d). To form new opcode replace [000] of RLCs with shown code.

8.8 Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
BIT b,r	ZF← $\overline{r_b}$	↕[1]	↕	↕[1]	1	↕[1]	↕[1]	0	•	11 001 011	CB	2	2	8	r Reg
										01 b r					000 B
BIT b,(HL)	ZF← $\overline{(HL)_b}$	↕[1]	↕	↕[1]	1	↕[1]	↕[1]	0	•	11 001 011	CB	2	3	12	001 C
										01 b 110					010 D
BIT b,(IX+d)[2]	ZF← $\overline{(IX + d)_b}$	↕[1]	↕	↕[1]	1	↕[1]	↕[1]	0	•	11 011 101	DD	4	5	20	011 E
										11 001 011	CB				100 H
										← d →					101 L
										01 b 110					111 A
BIT b,(IY+d)[2]	ZF← $\overline{(IY + d)_b}$	↕[1]	↕	↕[1]	1	↕[1]	↕[1]	0	•	11 111 101	FD	4	5	20	
										11 001 011	CB				
										← d →					
										01 b 110					
SET b,r	r_b ←1	•	•	•	•	•	•	•	•	11 001 011	CB	2	2	8	b Bit
										$\boxed{11}$ b r					000 0
SET b,(HL)	$(HL)_b$ ←1	•	•	•	•	•	•	•	•	11 001 011	CB	2	4	15	001 1
										$\boxed{11}$ b 110					010 2
SET b,(IX+d)	$(IX + d)_b$ ←1	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	23	011 3
										11 001 011	CB				100 4
										← d →					101 5
										$\boxed{11}$ b 110					110 6
SET b,(IY+d)	$(IY + d)_b$ ←1	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	23	111 7
										11 001 011	CB				
										← d →					
										$\boxed{11}$ b 110					
SET b,(IX+d),r	r←(IX+d)	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	23	
	r_b ←1									11 001 011	CB				
	(IX+d)←r									← d →					
										$\boxed{11}$ b r					
SET b,(IY+d),r	r←(IY+d)	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	23	
	r_b ←1									11 001 011	CB				
	(IY+d)←r									← d →					
										$\boxed{11}$ b r					
RES b,m	m_b ←0	•	•	•	•	•	•	•	•	$\boxed{10}$					

Note: [1]See section 4.1 for a complete description.
[2]Instruction has other undocumented opcodes.
m is one of r, (HL), (IX+d), (IY+d). To form RES instruction, replace $\boxed{11}$ with $\boxed{10}$.

8.9 Jump Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
JP nn	PC←nn	•	•	•	•	•	•	•	•	11 000 011	C3	3	3	10	cc Condition
										← n →					000 NZ
										← n →					001 Z
JP cc,nn	if cc	•	•	•	•	•	•	•	•	11 cc 010		3	3	10	010 NC
	PC←nn									← n →					011 C
										← n →					100 PO
JR e	PC←PC+e	•	•	•	•	•	•	•	•	00 011 000	18	2	3	12	101 PE
										← e-2 →					110 P
															111 M
JR ss,e	if ss	•	•	•	•	•	•	•	•	00 1ss 000		2	3	12	if ss is true
	PC←PC+e									← e-2 →		2	2	7	if ss is false
JP (HL)	PC←HL	•	•	•	•	•	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC←IX	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	8	ss Condition
										11 101 001	E9				11 C
JP (IY)	PC←IY	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	8	10 NC
										11 101 001	E9				01 Z
															00 NZ
DJNZ e	B←B-1	•	•	•	•	•	•	•	•	00 010 000	10	2	2	8	if B=0
	if B≠0									← e-2 →					
	PC←PC+e											2	3	13	if B≠0

Note: e is a signed two-compliment in the range -127, 129.
e-2 in the opcode provides an effective number of PC+e as PC is incremented by two prior to the addition of e.

8.10 Call and Return Group

	Symbolic	Flags								Opcode			M	T	
Mnemonic	Operation	SF	ZF	YF	HF	XF	PF	NF	CF	76 543 210	Hex	Bytes	Cycles	States	Comments
CALL nn	(SP-1)←PCh	•	•	•	•	•	•	•	•	11 001 101	CD	3	5	17	
	(SP-2)←PCl									← n →					
	SP←SP-2									← n →					
	PC←nn														
CALL cc,nn	if cc is true	•	•	•	•	•	•	•	•	11 cc 100		3	3	10	if cc is false
	(SP-1)←PCh									← n →		3	5	17	if cc is true
	(SP-2)←PCl									← n →					
	SP←SP-2														
	PC←nn														
RET	PCl←(SP)	•	•	•	•	•	•	•	•	11 001 001	C9	1	3	10	
	PCh←(SP+1)														
	SP←SP+2														
RET cc	if cc is true	•	•	•	•	•	•	•	•	11 cc 000		1	1	5	if cc is false
	PCl←(SP)											1	3	11	if cc is true
	PCh←(SP+1)														
	SP←SP+2														
RETI[1]	PCl←(SP)	•	•	•	•	•	•	•	•	11 101 101	ED	2	4	14	cc Condition
	PCh←(SP+1)									01 001 101	4D				000 NZ
	SP←SP+2														001 Z
	IFF1←IFF2														010 NC
RETN[2]	PCl←(SP)	•	•	•	•	•	•	•	•	11 101 101	ED	2	4	14	011 C
	PCh←(SP+1)									01 000 101	45				100 PO
	SP←SP+2														101 PE
	IFF1←IFF2														110 P
															111 M
RST p	(SP-1)←PCh	•	•	•	•	•	•	•	•	11 t 111		1	3	11	t p
	(SP-2)←PCl														000 0h
	SP←SP-2														001 8h
	PC←p														010 10h
															011 18h
															100 20h
															101 28h
															110 30h
															111 38h
Note:	[1] RETI also copies IFF2 into IFF1, like RETN. [2] This instruction has other undocumented opcodes.														

8.11 Input and Output Group

Mnemonic	Symbolic Operation	SF	ZF	YF	HF	XF	PF	NF	CF	Opcode 76 543 210	Hex	Bytes	M Cycles	T States	Comments
IN A,(n)	A←(n)	•	•	•	•	•	•	•	•	11 011 011 ← n →	DB	2	3	11	r Reg 000 B
IN r,(C)	r←(C)	↕	↕	↕	0	↕	PF	0	•	11 101 101 01 r 000	ED	2	3	12	001 C 010 D
IN F,(n)	←(C)	↕	↕	↕	0	↕	PF	0	•	11 101 101 01 110 000	ED 70	2	3	12	011 E 100 H
INI	(HL)←(C) HL←HL+1 B←B-1	↕[1]	↕[1]	↕[1]	↕[3]	↕[1]	↕[3]	↕[2]	↕[3]	11 101 101 10 100 010	ED A2	2	4	16	101 L 111 A
INIR	(HL)←(C) HL←HL+1 B←B-1 Repeat until B=0	0	1	0	↕[3]	0	↕[3]	↕[2]	↕[3]	11 101 101 10 110 010	ED B2	2 2	5 4	21 16	if B≠0 if B=0
IND	(HL)←(C) HL←HL-1 B←B-1	↕[1]	↕[1]	↕[1]	↕[3]	↕[1]	↕[3]	↕[2]	↕[4]	11 101 101 10 101 010	ED AA	2	4	16	
INDR	(HL)←(C) HL←HL-1 B←B-1 Repeat until B=0	0	1	0	↕[3]	0	↕[3]	↕[2]	↕[3]	11 101 101 10 111 010	ED BA	2 2	5 4	21 16	if B≠0 if B=0
OUT (n),A	(n)←A	•	•	•	•	•	•	•	•	11 010 011 ← n →	D3	2	3	11	
OUT (C),r	(C)←r	•	•	•	•	•	•	•	•	11 101 101 01 r 001	ED	2	3	12	
OUT (C),0	(C)←0	•	•	•	•	•	•	•	•	11 101 101 01 110 001	ED 71	2	3	12	
OUTI	(C)←(HL) HL←HL+1 B←B-1	↕[1]	↕[1]	↕[1]	↕[3]	↕[1]	↕[3]	↕[2]	↕[3]	11 101 101 10 100 011	ED A3	2	4	16	
OTIR	(C)←(HL) HL←HL+1 B←B-1 Repeat until B=0	0	1	0	↕[3]	0	↕	↕[2]	↕[3]	11 101 101 10 110 011	ED B3	2 2	5 4	21 16	if B≠0 if B=0
OUTD	(C)←(HL) HL←HL-1 B←B-1	↕[1]	↕[1]	↕[1]	↕[3]	↕[1]	↕[3]	↕[2]	↕[3]	11 101 101 10 101 011	ED AB	2	4	16	
OTDR	(C)←(HL) HL←HL-1 B←B-1 Repeat until B=0	0	1	0	↕[3]	0	↕[3]	↕[2]	↕[5]	11 101 101 10 111 011	ED BB	2 2	5 4	21 16	if B≠0 if B=0

Note:
[1] flag is affected by the result of B←B-1 as in DEC B.
[2] NF is a copy of bit 7 of the transferred byte.
[3] This flag is bizarre, see section 4.3.

Chapter 9

Instructions Sorted by Opcode

Any instruction marked with * is undocumented.

00	NOP
01 n n	LD BC,nn
02	LD (BC),A
03	INC BC
04	INC B
05	DEC B
06 n	LD B,n
07	RLCA
08	EX AF,AF’
09	ADD HL,BC
0A	LD A,(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E n	LD C,n
0F	RRCA
10 e	DJNZ (PC+e)
11 n n	LD DE,nn
12	LD (DE),A
13	INC DE
14	INC D
15	DEC D
16 n	LD D,n
17	RLA
18 e	JR e
19	ADD HL,DE
1A	LD A,(DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E n	LD E,n
1F	RRA
20 e	JR NZ,e
21 n n	LD HL,nn
22 n n	LD (nn),HL
23	INC HL
24	INC H
25	DEC H
26 n	LD H,n
27	DAA
28 e	JR Z,e
29	ADD HL,HL
2A n n	LD HL,(nn)
2B	DEC HL
2C	INC L
2D	DEC L
2E n	LD L,n
2F	CPL
30 e	JR NC,e
31 n n	LD SP,nn
32 n n	LD (nn),A
33	INC SP
34	INC (HL)
35	DEC (HL)
36 n	LD (HL),n
37	SCF
38 e	JR C,e
39	ADD HL,SP
3A n n	LD A,(nn)
3B	DEC SP
3C	INC A
3D	DEC A
3E n	LD A,n
3F	CCF
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
46	LD B,(HL)
47	LD B,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
4E	LD C,(HL)
4F	LD C,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
56	LD D,(HL)
57	LD D,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
5E	LD E,(HL)
5F	LD E,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
66	LD H,(HL)
67	LD H,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
6E	LD L,(HL)
6F	LD L,A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
7D	LD A,L
7E	LD A,(HL)
7F	LD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
86	ADD A,(HL)
87	ADD A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H

A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2 n n	JP NZ,nn
C3 n n	JP nn
C4 n n	CALL NZ,nn
C5	PUSH BC
C6 n	ADD A,n
C7	RST 0H
C8	RET Z
C9	RET
CA n n	JP Z,nn
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB26	SLA (HL)
CB27	SLA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRA (HL)
CB2F	SRA A
CB30	SLL B*
CB31	SLL C*
CB32	SLL D*
CB33	SLL E*
CB34	SLL H*
CB35	SLL L*
CB36	SLL (HL)*
CB37	SLL A*
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
CB3E	SRL (HL)
CB3F	SRL A
CB40	BIT 0,B
CB41	BIT 0,C
CB42	BIT 0,D
CB43	BIT 0,E
CB44	BIT 0,H
CB45	BIT 0,L
CB46	BIT 0,(HL)
CB47	BIT 0,A
CB48	BIT 1,B
CB49	BIT 1,C
CB4A	BIT 1,D
CB4B	BIT 1,E
CB4C	BIT 1,H
CB4D	BIT 1,L
CB4E	BIT 1,(HL)
CB4F	BIT 1,A
CB50	BIT 2,B
CB51	BIT 2,C
CB52	BIT 2,D
CB53	BIT 2,E
CB54	BIT 2,H
CB55	BIT 2,L
CB56	BIT 2,(HL)
CB57	BIT 2,A
CB58	BIT 3,B
CB59	BIT 3,C
CB5A	BIT 3,D
CB5B	BIT 3,E
CB5C	BIT 3,H
CB5D	BIT 3,L
CB5E	BIT 3,(HL)
CB5F	BIT 3,A
CB60	BIT 4,B
CB61	BIT 4,C
CB62	BIT 4,D
CB63	BIT 4,E
CB64	BIT 4,H
CB65	BIT 4,L
CB66	BIT 4,(HL)
CB67	BIT 4,A
CB68	BIT 5,B
CB69	BIT 5,C
CB6A	BIT 5,D
CB6B	BIT 5,E
CB6C	BIT 5,H
CB6D	BIT 5,L
CB6E	BIT 5,(HL)
CB6F	BIT 5,A
CB70	BIT 6,B
CB71	BIT 6,C
CB72	BIT 6,D
CB73	BIT 6,E
CB74	BIT 6,H
CB75	BIT 6,L
CB76	BIT 6,(HL)
CB77	BIT 6,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
CB7E	BIT 7,(HL)
CB7F	BIT 7,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H
CB85	RES 0,L
CB86	RES 0,(HL)
CB87	RES 0,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB8E	RES 1,(HL)
CB8F	RES 1,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB96	RES 2,(HL)
CB97	RES 2,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CB9E	RES 3,(HL)
CB9F	RES 3,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBA6	RES 4,(HL)
CBA7	RES 4,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBAE	RES 5,(HL)
CBAF	RES 5,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBB6	RES 6,(HL)
CBB7	RES 6,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
CBBE	RES 7,(HL)
CBBF	RES 7,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBC6	SET 0,(HL)
CBC7	SET 0,A
CBC8	SET 1,B
CBC9	SET 1,C

CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBCE	SET 1,(HL)
CBCF	SET 1,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBE6	SET 4,(HL)
CBE7	SET 4,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBEE	SET 5,(HL)
CBEF	SET 5,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBF6	SET 6,(HL)
CBF7	SET 6,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CBFE	SET 7,(HL)
CBFF	SET 7,A
CC n n	CALL Z,nn
CD n n	CALL nn
CE n	ADC A,n
CF	RST 8H
D0	RET NC
D1	POP DE
D2 n n	JP NC,nn
D3 n	OUT (n),A
D4 n n	CALL NC,nn
D5	PUSH DE
D6 n	SUB n
D7	RST 10H
D8	RET C
D9	EXX
DA n n	JP C,nn
DB n	IN A,(n)
DC n n	CALL C,nn
DD09	ADD IX,BC
DD19	ADD IX,DE
DD21 n n	LD IX,nn
DD22 n n	LD (nn),IX
DD23	INC IX
DD24	INC IXh*
DD25	DEC IXh*
DD26 n	LD IXh,n*
DD29	ADD IX,IX
DD2A n n	LD IX,(nn)
DD2B	DEC IX
DD2C	INC IXl*
DD2D	DEC IXl*
DD2E n	LD IXl,n*
DD34 d	INC (IX+d)
DD35 d	DEC (IX+d)
DD36 d n	LD (IX+d),n
DD39	ADD IX,SP
DD44	LD B,IXh*
DD45	LD B,IXl*
DD46 d	LD B,(IX+d)
DD4C	LD C,IXh*
DD4D	LD C,IXl*
DD4E d	LD C,(IX+d)
DD54	LD D,IXh*
DD55	LD D,IXl*
DD56 d	LD D,(IX+d)
DD5C	LD E,IXh*
DD5D	LD E,IXl*
DD5E d	LD E,(IX+d)
DD60	LD IXh,B*
DD61	LD IXh,C*
DD62	LD IXh,D*
DD63	LD IXh,E*
DD64	LD IXh,IXh*
DD65	LD IXh,IXl*
DD66 d	LD H,(IX+d)
DD67	LD IXh,A*
DD68	LD IXl,B*
DD69	LD IXl,C*
DD6A	LD IXl,D*
DD6B	LD IXl,E*
DD6C	LD IXl,IXh*
DD6D	LD IXl,IXl*
DD6E d	LD L,(IX+d)
DD6F	LD IXl,A*
DD70 d	LD (IX+d),B
DD71 d	LD (IX+d),C
DD72 d	LD (IX+d),D
DD73 d	LD (IX+d),E
DD74 d	LD (IX+d),H
DD75 d	LD (IX+d),L
DD77 d	LD (IX+d),A
DD7C	LD A,IXh*
DD7D	LD A,IXl*
DD7E d	LD A,(IX+d)
DD84	ADD A,IXh*
DD85	ADD A,IXl*
DD86 d	ADD A,(IX+d)
DD8C	ADC A,IXh*
DD8D	ADC A,IXl*
DD8E d	ADC A,(IX+d)
DD94	SUB IXh*
DD95	SUB IXl*
DD96 d	SUB (IX+d)
DD9C	SBC A,IXh*
DD9D	SBC A,IXl*
DD9E d	SBC A,(IX+d)
DDA4	AND IXh*
DDA5	AND IXl*
DDA6 d	AND (IX+d)
DDAC	XOR IXh*
DDAD	XOR IXl*
DDAE d	XOR (IX+d)
DDB4	OR IXh*
DDB5	OR IXl*
DDB6 d	OR (IX+d)
DDBC	CP IXh*
DDBD	CP IXl*
DDBE d	CP (IX+d)
DDCB d 00	RLC (IX+d),B*
DDCB d 01	RLC (IX+d),C*
DDCB d 02	RLC (IX+d),D*
DDCB d 03	RLC (IX+d),E*
DDCB d 04	RLC (IX+d),H*
DDCB d 05	RLC (IX+d),L*
DDCB d 06	RLC (IX+d)
DDCB d 07	RLC (IX+d),A*
DDCB d 08	RRC (IX+d),B*
DDCB d 09	RRC (IX+d),C*
DDCB d 0A	RRC (IX+d),D*
DDCB d 0B	RRC (IX+d),E*
DDCB d 0C	RRC (IX+d),H*
DDCB d 0D	RRC (IX+d),L*
DDCB d 0E	RRC (IX+d)
DDCB d 0F	RRC (IX+d),A*
DDCB d 10	RL (IX+d),B*
DDCB d 11	RL (IX+d),C*
DDCB d 12	RL (IX+d),D*
DDCB d 13	RL (IX+d),E*
DDCB d 14	RL (IX+d),H*
DDCB d 15	RL (IX+d),L*
DDCB d 16	RL (IX+d)
DDCB d 17	RL (IX+d),A*
DDCB d 18	RR (IX+d),B*
DDCB d 19	RR (IX+d),C*
DDCB d 1A	RR (IX+d),D*
DDCB d 1B	RR (IX+d),E*
DDCB d 1C	RR (IX+d),H*
DDCB d 1D	RR (IX+d),L*
DDCB d 1E	RR (IX+d)
DDCB d 1F	RR (IX+d),A*
DDCB d 20	SLA (IX+d),B*
DDCB d 21	SLA (IX+d),C*
DDCB d 22	SLA (IX+d),D*
DDCB d 23	SLA (IX+d),E*
DDCB d 24	SLA (IX+d),H*
DDCB d 25	SLA (IX+d),L*
DDCB d 26	SLA (IX+d)
DDCB d 27	SLA (IX+d),A*
DDCB d 28	SRA (IX+d),B*
DDCB d 29	SRA (IX+d),C*
DDCB d 2A	SRA (IX+d),D*
DDCB d 2B	SRA (IX+d),E*
DDCB d 2C	SRA (IX+d),H*
DDCB d 2D	SRA (IX+d),L*
DDCB d 2E	SRA (IX+d)
DDCB d 2F	SRA (IX+d),A*
DDCB d 30	SLL (IX+d),B*
DDCB d 31	SLL (IX+d),C*
DDCB d 32	SLL (IX+d),D*
DDCB d 33	SLL (IX+d),E*
DDCB d 34	SLL (IX+d),H*
DDCB d 35	SLL (IX+d),L*
DDCB d 36	SLL (IX+d)*
DDCB d 37	SLL (IX+d),A*
DDCB d 38	SRL (IX+d),B*
DDCB d 39	SRL (IX+d),C*
DDCB d 3A	SRL (IX+d),D*
DDCB d 3B	SRL (IX+d),E*
DDCB d 3C	SRL (IX+d),H*
DDCB d 3D	SRL (IX+d),L*
DDCB d 3E	SRL (IX+d)
DDCB d 3F	SRL (IX+d),A*
DDCB d 40	BIT 0,(IX+d)*
DDCB d 41	BIT 0,(IX+d)*
DDCB d 42	BIT 0,(IX+d)*
DDCB d 43	BIT 0,(IX+d)*
DDCB d 44	BIT 0,(IX+d)*
DDCB d 45	BIT 0,(IX+d)*
DDCB d 46	BIT 0,(IX+d)
DDCB d 47	BIT 0,(IX+d)*
DDCB d 48	BIT 1,(IX+d)*
DDCB d 49	BIT 1,(IX+d)*
DDCB d 4A	BIT 1,(IX+d)*
DDCB d 4B	BIT 1,(IX+d)*
DDCB d 4C	BIT 1,(IX+d)*
DDCB d 4D	BIT 1,(IX+d)*
DDCB d 4E	BIT 1,(IX+d)
DDCB d 4F	BIT 1,(IX+d)*
DDCB d 50	BIT 2,(IX+d)*
DDCB d 51	BIT 2,(IX+d)*
DDCB d 52	BIT 2,(IX+d)*
DDCB d 53	BIT 2,(IX+d)*
DDCB d 54	BIT 2,(IX+d)*
DDCB d 55	BIT 2,(IX+d)*
DDCB d 56	BIT 2,(IX+d)
DDCB d 57	BIT 2,(IX+d)*
DDCB d 58	BIT 3,(IX+d)*

DDCB d 59	BIT 3,(IX+d)*
DDCB d 5A	BIT 3,(IX+d)*
DDCB d 5B	BIT 3,(IX+d)*
DDCB d 5C	BIT 3,(IX+d)*
DDCB d 5D	BIT 3,(IX+d)*
DDCB d 5E	BIT 3,(IX+d)
DDCB d 5F	BIT 3,(IX+d)*
DDCB d 60	BIT 4,(IX+d)*
DDCB d 61	BIT 4,(IX+d)*
DDCB d 62	BIT 4,(IX+d)*
DDCB d 63	BIT 4,(IX+d)*
DDCB d 64	BIT 4,(IX+d)*
DDCB d 65	BIT 4,(IX+d)*
DDCB d 66	BIT 4,(IX+d)
DDCB d 67	BIT 4,(IX+d)*
DDCB d 68	BIT 5,(IX+d)*
DDCB d 69	BIT 5,(IX+d)*
DDCB d 6A	BIT 5,(IX+d)*
DDCB d 6B	BIT 5,(IX+d)*
DDCB d 6C	BIT 5,(IX+d)*
DDCB d 6D	BIT 5,(IX+d)*
DDCB d 6E	BIT 5,(IX+d)
DDCB d 6F	BIT 5,(IX+d)*
DDCB d 70	BIT 6,(IX+d)*
DDCB d 71	BIT 6,(IX+d)*
DDCB d 72	BIT 6,(IX+d)*
DDCB d 73	BIT 6,(IX+d)*
DDCB d 74	BIT 6,(IX+d)*
DDCB d 75	BIT 6,(IX+d)*
DDCB d 76	BIT 6,(IX+d)
DDCB d 77	BIT 6,(IX+d)*
DDCB d 78	BIT 7,(IX+d)*
DDCB d 79	BIT 7,(IX+d)*
DDCB d 7A	BIT 7,(IX+d)*
DDCB d 7B	BIT 7,(IX+d)*
DDCB d 7C	BIT 7,(IX+d)*
DDCB d 7D	BIT 7,(IX+d)*
DDCB d 7E	BIT 7,(IX+d)
DDCB d 7F	BIT 7,(IX+d)*
DDCB d 80	RES 0,(IX+d),B*
DDCB d 81	RES 0,(IX+d),C*
DDCB d 82	RES 0,(IX+d),D*
DDCB d 83	RES 0,(IX+d),E*
DDCB d 84	RES 0,(IX+d),H*
DDCB d 85	RES 0,(IX+d),L*
DDCB d 86	RES 0,(IX+d)
DDCB d 87	RES 0,(IX+d),A*
DDCB d 88	RES 1,(IX+d),B*
DDCB d 89	RES 1,(IX+d),C*
DDCB d 8A	RES 1,(IX+d),D*
DDCB d 8B	RES 1,(IX+d),E*
DDCB d 8C	RES 1,(IX+d),H*
DDCB d 8D	RES 1,(IX+d),L*
DDCB d 8E	RES 1,(IX+d)
DDCB d 8F	RES 1,(IX+d),A*
DDCB d 90	RES 2,(IX+d),B*
DDCB d 91	RES 2,(IX+d),C*
DDCB d 92	RES 2,(IX+d),D*
DDCB d 93	RES 2,(IX+d),E*
DDCB d 94	RES 2,(IX+d),H*
DDCB d 95	RES 2,(IX+d),L*
DDCB d 96	RES 2,(IX+d)
DDCB d 97	RES 2,(IX+d),A*
DDCB d 98	RES 3,(IX+d),B*
DDCB d 99	RES 3,(IX+d),C*
DDCB d 9A	RES 3,(IX+d),D*
DDCB d 9B	RES 3,(IX+d),E*
DDCB d 9C	RES 3,(IX+d),H*
DDCB d 9D	RES 3,(IX+d),L*
DDCB d 9E	RES 3,(IX+d)
DDCB d 9F	RES 3,(IX+d),A*
DDCB d A0	RES 4,(IX+d),B*
DDCB d A1	RES 4,(IX+d),C*
DDCB d A2	RES 4,(IX+d),D*
DDCB d A3	RES 4,(IX+d),E*
DDCB d A4	RES 4,(IX+d),H*
DDCB d A5	RES 4,(IX+d),L*
DDCB d A6	RES 4,(IX+d)
DDCB d A7	RES 4,(IX+d),A*
DDCB d A8	RES 5,(IX+d),B*
DDCB d A9	RES 5,(IX+d),C*
DDCB d AA	RES 5,(IX+d),D*
DDCB d AB	RES 5,(IX+d),E*
DDCB d AC	RES 5,(IX+d),H*
DDCB d AD	RES 5,(IX+d),L*
DDCB d AE	RES 5,(IX+d)
DDCB d AF	RES 5,(IX+d),A*
DDCB d B0	RES 6,(IX+d),B*
DDCB d B1	RES 6,(IX+d),C*
DDCB d B2	RES 6,(IX+d),D*
DDCB d B3	RES 6,(IX+d),E*
DDCB d B4	RES 6,(IX+d),H*
DDCB d B5	RES 6,(IX+d),L*
DDCB d B6	RES 6,(IX+d)
DDCB d B7	RES 6,(IX+d),A*
DDCB d B8	RES 7,(IX+d),B*
DDCB d B9	RES 7,(IX+d),C*
DDCB d BA	RES 7,(IX+d),D*
DDCB d BB	RES 7,(IX+d),E*
DDCB d BC	RES 7,(IX+d),H*
DDCB d BD	RES 7,(IX+d),L*
DDCB d BE	RES 7,(IX+d)
DDCB d BF	RES 7,(IX+d),A*
DDCB d C0	SET 0,(IX+d),B*
DDCB d C1	SET 0,(IX+d),C*
DDCB d C2	SET 0,(IX+d),D*
DDCB d C3	SET 0,(IX+d),E*
DDCB d C4	SET 0,(IX+d),H*
DDCB d C5	SET 0,(IX+d),L*
DDCB d C6	SET 0,(IX+d)
DDCB d C7	SET 0,(IX+d),A*
DDCB d C8	SET 1,(IX+d),B*
DDCB d C9	SET 1,(IX+d),C*
DDCB d CA	SET 1,(IX+d),D*
DDCB d CB	SET 1,(IX+d),E*
DDCB d CC	SET 1,(IX+d),H*
DDCB d CD	SET 1,(IX+d),L*
DDCB d CE	SET 1,(IX+d)
DDCB d CF	SET 1,(IX+d),A*
DDCB d D0	SET 2,(IX+d),B*
DDCB d D1	SET 2,(IX+d),C*
DDCB d D2	SET 2,(IX+d),D*
DDCB d D3	SET 2,(IX+d),E*
DDCB d D4	SET 2,(IX+d),H*
DDCB d D5	SET 2,(IX+d),L*
DDCB d D6	SET 2,(IX+d)
DDCB d D7	SET 2,(IX+d),A*
DDCB d D8	SET 3,(IX+d),B*
DDCB d D9	SET 3,(IX+d),C*
DDCB d DA	SET 3,(IX+d),D*
DDCB d DB	SET 3,(IX+d),E*
DDCB d DC	SET 3,(IX+d),H*
DDCB d DD	SET 3,(IX+d),L*
DDCB d DE	SET 3,(IX+d)
DDCB d DF	SET 3,(IX+d),A*
DDCB d E0	SET 4,(IX+d),B*
DDCB d E1	SET 4,(IX+d),C*
DDCB d E2	SET 4,(IX+d),D*
DDCB d E3	SET 4,(IX+d),E*
DDCB d E4	SET 4,(IX+d),H*
DDCB d E5	SET 4,(IX+d),L*
DDCB d E6	SET 4,(IX+d)
DDCB d E7	SET 4,(IX+d),A*
DDCB d E8	SET 5,(IX+d),B*
DDCB d E9	SET 5,(IX+d),C*
DDCB d EA	SET 5,(IX+d),D*
DDCB d EB	SET 5,(IX+d),E*
DDCB d EC	SET 5,(IX+d),H*
DDCB d ED	SET 5,(IX+d),L*
DDCB d EE	SET 5,(IX+d)
DDCB d EF	SET 5,(IX+d),A*
DDCB d F0	SET 6,(IX+d),B*
DDCB d F1	SET 6,(IX+d),C*
DDCB d F2	SET 6,(IX+d),D*
DDCB d F3	SET 6,(IX+d),E*
DDCB d F4	SET 6,(IX+d),H*
DDCB d F5	SET 6,(IX+d),L*
DDCB d F6	SET 6,(IX+d)
DDCB d F7	SET 6,(IX+d),A*
DDCB d F8	SET 7,(IX+d),B*
DDCB d F9	SET 7,(IX+d),C*
DDCB d FA	SET 7,(IX+d),D*
DDCB d FB	SET 7,(IX+d),E*
DDCB d FC	SET 7,(IX+d),H*
DDCB d FD	SET 7,(IX+d),L*
DDCB d FE	SET 7,(IX+d)
DDCB d FF	SET 7,(IX+d),A*
DDE1	POP IX
DDE3	EX (SP),IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP,IX
DE n	SBC A,n
DF	RST 18H
E0	RET PO
E1	POP HL
E2 n n	JP PO,nn
E3	EX (SP),HL
E4 n n	CALL PO,nn
E5	PUSH HL
E6 n	AND n
E7	RST 20H
E8	RET PE
E9	JP (HL)
EA n n	JP PE,nn
EB	EX DE,HL
EC n n	CALL PE,nn
ED40	IN B,(C)
ED41	OUT (C),B
ED42	SBC HL,BC
ED43 n n	LD (nn),BC
ED44	NEG
ED45	RETN
ED46	IM 0
ED47	LD I,A
ED48	IN C,(C)
ED49	OUT (C),C
ED4A	ADC HL,BC
ED4B n n	LD BC,(nn)
ED4C	NEG*
ED4D	RETI
ED4E	IM 0*
ED4F	LD R,A
ED50	IN D,(C)
ED51	OUT (C),D
ED52	SBC HL,DE
ED53 n n	LD (nn),DE
ED54	NEG*
ED55	RETN*
ED56	IM 1
ED57	LD A,I
ED58	IN E,(C)
ED59	OUT (C),E
ED5A	ADC HL,DE
ED5B n n	LD DE,(nn)
ED5C	NEG*
ED5D	RETN*
ED5E	IM 2
ED5F	LD A,R
ED60	IN H,(C)
ED61	OUT (C),H
ED62	SBC HL,HL
ED63 n n	LD (nn),HL
ED64	NEG*
ED65	RETN*
ED66	IM 0*
ED67	RRD
ED68	IN L,(C)
ED69	OUT (C),L
ED6A	ADC HL,HL
ED6B n n	LD HL,(nn)
ED6C	NEG*
ED6D	RETN*
ED6E	IM 0*
ED6F	RLD
ED70	IN F,(C)* / IN (C)*
ED71	OUT (C),0*
ED72	SBC HL,SP
ED73 n n	LD (nn),SP
ED74	NEG*

ED75	RETN*
ED76	IM 1*
ED78	IN A,(C)
ED79	OUT (C),A
ED7A	ADC HL,SP
ED7B n n	LD SP,(nn)
ED7C	NEG*
ED7D	RETN*
ED7E	IM 2*
EDA0	LDI
EDA1	CPI
EDA2	INI
EDA3	OUTI
EDA8	LDD
EDA9	CPD
EDAA	IND
EDAB	OUTD
EDB0	LDIR
EDB1	CPIR
EDB2	INIR
EDB3	OTIR
EDB8	LDDR
EDB9	CPDR
EDBA	INDR
EDBB	OTDR
EE n	XOR n
EF	RST 28H
F0	RET P
F1	POP AF
F2 n n	JP P,nn
F3	DI
F4 n n	CALL P,nn
F5	PUSH AF
F6 n	OR n
F7	RST 30H
F8	RET M
F9	LD SP,HL
FA n n	JP M,nn
FB	EI
FC n n	CALL M,nn
FD09	ADD IY,BC
FD19	ADD IY,DE
FD21 n n	LD IY,nn
FD22 n n	LD (nn),IY
FD23	INC IY
FD24	INC IYh*
FD25	DEC IYh*
FD26 n	LD IYh,n*
FD29	ADD IY,IY
FD2A n n	LD IY,(nn)
FD2B	DEC IY
FD2C	INC IYl*
FD2D	DEC IYl*
FD2E n	LD IYl,n*
FD34 d	INC (IY+d)
FD35 d	DEC (IY+d)
FD36 d n	LD (IY+d),n
FD39	ADD IY,SP
FD44	LD B,IYh*
FD45	LD B,IYl*
FD46 d	LD B,(IY+d)
FD4C	LD C,IYh*
FD4D	LD C,IYl*
FD4E d	LD C,(IY+d)
FD54	LD D,IYh*
FD55	LD D,IYl*
FD56 d	LD D,(IY+d)
FD5C	LD E,IYh*
FD5D	LD E,IYl*
FD5E d	LD E,(IY+d)
FD60	LD IYh,B*
FD61	LD IYh,C*
FD62	LD IYh,D*
FD63	LD IYh,E*
FD64	LD IYh,IYh*
FD65	LD IYh,IYl*
FD66 d	LD H,(IY+d)
FD67	LD IYh,A*
FD68	LD IYl,B*
FD69	LD IYl,C*
FD6A	LD IYl,D*
FD6B	LD IYl,E*
FD6C	LD IYl,IYh*
FD6D	LD IYl,IYl*
FD6E d	LD L,(IY+d)
FD6F	LD IYl,A*
FD70 d	LD (IY+d),B
FD71 d	LD (IY+d),C
FD72 d	LD (IY+d),D
FD73 d	LD (IY+d),E
FD74 d	LD (IY+d),H
FD75 d	LD (IY+d),L
FD77 d	LD (IY+d),A
FD7C	LD A,IYh*
FD7D	LD A,IYl*
FD7E d	LD A,(IY+d)
FD84	ADD A,IYh*
FD85	ADD A,IYl*
FD86 d	ADD A,(IY+d)
FD8C	ADC A,IYh*
FD8D	ADC A,IYl*
FD8E d	ADC A,(IY+d)
FD94	SUB IYh*
FD95	SUB IYl*
FD96 d	SUB (IY+d)
FD9C	SBC A,IYh*
FD9D	SBC A,IYl*
FD9E d	SBC A,(IY+d)
FDA4	AND IYh*
FDA5	AND IYl*
FDA6 d	AND (IY+d)
FDAC	XOR IYh*
FDAD	XOR IYl*
FDAE d	XOR (IY+d)
FDB4	OR IYh*
FDB5	OR IYl*
FDB6 d	OR (IY+d)
FDBC	CP IYh*
FDBD	CP IYl*
FDBE d	CP (IY+d)
FDCB d 00	RLC (IY+d),B*
FDCB d 01	RLC (IY+d),C*
FDCB d 02	RLC (IY+d),D*
FDCB d 03	RLC (IY+d),E*
FDCB d 04	RLC (IY+d),H*
FDCB d 05	RLC (IY+d),L*
FDCB d 06	RLC (IY+d)
FDCB d 07	RLC (IY+d),A*
FDCB d 08	RRC (IY+d),B*
FDCB d 09	RRC (IY+d),C*
FDCB d 0A	RRC (IY+d),D*
FDCB d 0B	RRC (IY+d),E*
FDCB d 0C	RRC (IY+d),H*
FDCB d 0D	RRC (IY+d),L*
FDCB d 0E	RRC (IY+d)
FDCB d 0F	RRC (IY+d),A*
FDCB d 10	RL (IY+d),B*
FDCB d 11	RL (IY+d),C*
FDCB d 12	RL (IY+d),D*
FDCB d 13	RL (IY+d),E*
FDCB d 14	RL (IY+d),H*
FDCB d 15	RL (IY+d),L*
FDCB d 16	RL (IY+d)
FDCB d 17	RL (IY+d),A*
FDCB d 18	RR (IY+d),B*
FDCB d 19	RR (IY+d),C*
FDCB d 1A	RR (IY+d),D*
FDCB d 1B	RR (IY+d),E*
FDCB d 1C	RR (IY+d),H*
FDCB d 1D	RR (IY+d),L*
FDCB d 1E	RR (IY+d)
FDCB d 1F	RR (IY+d),A*
FDCB d 20	SLA (IY+d),B*
FDCB d 21	SLA (IY+d),C*
FDCB d 22	SLA (IY+d),D*
FDCB d 23	SLA (IY+d),E*
FDCB d 24	SLA (IY+d),H*
FDCB d 25	SLA (IY+d),L*
FDCB d 26	SLA (IY+d)
FDCB d 27	SLA (IY+d),A*
FDCB d 28	SRA (IY+d),B*
FDCB d 29	SRA (IY+d),C*
FDCB d 2A	SRA (IY+d),D*
FDCB d 2B	SRA (IY+d),E*
FDCB d 2C	SRA (IY+d),H*
FDCB d 2D	SRA (IY+d),L*
FDCB d 2E	SRA (IY+d)
FDCB d 2F	SRA (IY+d),A*
FDCB d 30	SLL (IY+d),B*
FDCB d 31	SLL (IY+d),C*
FDCB d 32	SLL (IY+d),D*
FDCB d 33	SLL (IY+d),E*
FDCB d 34	SLL (IY+d),H*
FDCB d 35	SLL (IY+d),L*
FDCB d 36	SLL (IY+d)*
FDCB d 37	SLL (IY+d),A*
FDCB d 38	SRL (IY+d),B*
FDCB d 39	SRL (IY+d),C*
FDCB d 3A	SRL (IY+d),D*
FDCB d 3B	SRL (IY+d),E*
FDCB d 3C	SRL (IY+d),H*
FDCB d 3D	SRL (IY+d),L*
FDCB d 3E	SRL (IY+d)
FDCB d 3F	SRL (IY+d),A*
FDCB d 40	BIT 0,(IY+d)*
FDCB d 41	BIT 0,(IY+d)*
FDCB d 42	BIT 0,(IY+d)*
FDCB d 43	BIT 0,(IY+d)*
FDCB d 44	BIT 0,(IY+d)*
FDCB d 45	BIT 0,(IY+d)*
FDCB d 46	BIT 0,(IY+d)
FDCB d 47	BIT 0,(IY+d)*
FDCB d 48	BIT 1,(IY+d)*
FDCB d 49	BIT 1,(IY+d)*
FDCB d 4A	BIT 1,(IY+d)*
FDCB d 4B	BIT 1,(IY+d)*
FDCB d 4C	BIT 1,(IY+d)*
FDCB d 4D	BIT 1,(IY+d)*
FDCB d 4E	BIT 1,(IY+d)
FDCB d 4F	BIT 1,(IY+d)*
FDCB d 50	BIT 2,(IY+d)*
FDCB d 51	BIT 2,(IY+d)*
FDCB d 52	BIT 2,(IY+d)*
FDCB d 53	BIT 2,(IY+d)*
FDCB d 54	BIT 2,(IY+d)*
FDCB d 55	BIT 2,(IY+d)*
FDCB d 56	BIT 2,(IY+d)
FDCB d 57	BIT 2,(IY+d)*
FDCB d 58	BIT 3,(IY+d)*
FDCB d 59	BIT 3,(IY+d)*
FDCB d 5A	BIT 3,(IY+d)*
FDCB d 5B	BIT 3,(IY+d)*
FDCB d 5C	BIT 3,(IY+d)*
FDCB d 5D	BIT 3,(IY+d)*
FDCB d 5E	BIT 3,(IY+d)
FDCB d 5F	BIT 3,(IY+d)*
FDCB d 60	BIT 4,(IY+d)*
FDCB d 61	BIT 4,(IY+d)*
FDCB d 62	BIT 4,(IY+d)*
FDCB d 63	BIT 4,(IY+d)*
FDCB d 64	BIT 4,(IY+d)*
FDCB d 65	BIT 4,(IY+d)*
FDCB d 66	BIT 4,(IY+d)
FDCB d 67	BIT 4,(IY+d)*
FDCB d 68	BIT 5,(IY+d)*
FDCB d 69	BIT 5,(IY+d)*
FDCB d 6A	BIT 5,(IY+d)*
FDCB d 6B	BIT 5,(IY+d)*
FDCB d 6C	BIT 5,(IY+d)*
FDCB d 6D	BIT 5,(IY+d)*
FDCB d 6E	BIT 5,(IY+d)
FDCB d 6F	BIT 5,(IY+d)*
FDCB d 70	BIT 6,(IY+d)*
FDCB d 71	BIT 6,(IY+d)*
FDCB d 72	BIT 6,(IY+d)*
FDCB d 73	BIT 6,(IY+d)*
FDCB d 74	BIT 6,(IY+d)*
FDCB d 75	BIT 6,(IY+d)*
FDCB d 76	BIT 6,(IY+d)
FDCB d 77	BIT 6,(IY+d)*

FDCB d 78	BIT 7,(IY+d)*
FDCB d 79	BIT 7,(IY+d)*
FDCB d 7A	BIT 7,(IY+d)*
FDCB d 7B	BIT 7,(IY+d)*
FDCB d 7C	BIT 7,(IY+d)*
FDCB d 7D	BIT 7,(IY+d)*
FDCB d 7E	BIT 7,(IY+d)
FDCB d 7F	BIT 7,(IY+d)*
FDCB d 80	RES 0,(IY+d),B*
FDCB d 81	RES 0,(IY+d),C*
FDCB d 82	RES 0,(IY+d),D*
FDCB d 83	RES 0,(IY+d),E*
FDCB d 84	RES 0,(IY+d),H*
FDCB d 85	RES 0,(IY+d),L*
FDCB d 86	RES 0,(IY+d)
FDCB d 87	RES 0,(IY+d),A*
FDCB d 88	RES 1,(IY+d),B*
FDCB d 89	RES 1,(IY+d),C*
FDCB d 8A	RES 1,(IY+d),D*
FDCB d 8B	RES 1,(IY+d),E*
FDCB d 8C	RES 1,(IY+d),H*
FDCB d 8D	RES 1,(IY+d),L*
FDCB d 8E	RES 1,(IY+d)
FDCB d 8F	RES 1,(IY+d),A*
FDCB d 90	RES 2,(IY+d),B*
FDCB d 91	RES 2,(IY+d),C*
FDCB d 92	RES 2,(IY+d),D*
FDCB d 93	RES 2,(IY+d),E*
FDCB d 94	RES 2,(IY+d),H*
FDCB d 95	RES 2,(IY+d),L*
FDCB d 96	RES 2,(IY+d)
FDCB d 97	RES 2,(IY+d),A*
FDCB d 98	RES 3,(IY+d),B*
FDCB d 99	RES 3,(IY+d),C*
FDCB d 9A	RES 3,(IY+d),D*
FDCB d 9B	RES 3,(IY+d),E*
FDCB d 9C	RES 3,(IY+d),H*
FDCB d 9D	RES 3,(IY+d),L*
FDCB d 9E	RES 3,(IY+d)
FDCB d 9F	RES 3,(IY+d),A*
FDCB d A0	RES 4,(IY+d),B*
FDCB d A1	RES 4,(IY+d),C*
FDCB d A2	RES 4,(IY+d),D*
FDCB d A3	RES 4,(IY+d),E*
FDCB d A4	RES 4,(IY+d),H*
FDCB d A5	RES 4,(IY+d),L*
FDCB d A6	RES 4,(IY+d)
FDCB d A7	RES 4,(IY+d),A*
FDCB d A8	RES 5,(IY+d),B*
FDCB d A9	RES 5,(IY+d),C*
FDCB d AA	RES 5,(IY+d),D*
FDCB d AB	RES 5,(IY+d),E*
FDCB d AC	RES 5,(IY+d),H*
FDCB d AD	RES 5,(IY+d),L*
FDCB d AE	RES 5,(IY+d)
FDCB d AF	RES 5,(IY+d),A*
FDCB d B0	RES 6,(IY+d),B*
FDCB d B1	RES 6,(IY+d),C*
FDCB d B2	RES 6,(IY+d),D*
FDCB d B3	RES 6,(IY+d),E*
FDCB d B4	RES 6,(IY+d),H*
FDCB d B5	RES 6,(IY+d),L*
FDCB d B6	RES 6,(IY+d)
FDCB d B7	RES 6,(IY+d),A*
FDCB d B8	RES 7,(IY+d),B*
FDCB d B9	RES 7,(IY+d),C*
FDCB d BA	RES 7,(IY+d),D*
FDCB d BB	RES 7,(IY+d),E*
FDCB d BC	RES 7,(IY+d),H*
FDCB d BD	RES 7,(IY+d),L*
FDCB d BE	RES 7,(IY+d)
FDCB d BF	RES 7,(IY+d),A*
FDCB d C0	SET 0,(IY+d),B*
FDCB d C1	SET 0,(IY+d),C*
FDCB d C2	SET 0,(IY+d),D*
FDCB d C3	SET 0,(IY+d),E*
FDCB d C4	SET 0,(IY+d),H*
FDCB d C5	SET 0,(IY+d),L*
FDCB d C6	SET 0,(IY+d)
FDCB d C7	SET 0,(IY+d),A*
FDCB d C8	SET 1,(IY+d),B*
FDCB d C9	SET 1,(IY+d),C*
FDCB d CA	SET 1,(IY+d),D*
FDCB d CB	SET 1,(IY+d),E*
FDCB d CC	SET 1,(IY+d),H*
FDCB d CD	SET 1,(IY+d),L*
FDCB d CE	SET 1,(IY+d)
FDCB d CF	SET 1,(IY+d),A*
FDCB d D0	SET 2,(IY+d),B*
FDCB d D1	SET 2,(IY+d),C*
FDCB d D2	SET 2,(IY+d),D*
FDCB d D3	SET 2,(IY+d),E*
FDCB d D4	SET 2,(IY+d),H*
FDCB d D5	SET 2,(IY+d),L*
FDCB d D6	SET 2,(IY+d)
FDCB d D7	SET 2,(IY+d),A*
FDCB d D8	SET 3,(IY+d),B*
FDCB d D9	SET 3,(IY+d),C*
FDCB d DA	SET 3,(IY+d),D*
FDCB d DB	SET 3,(IY+d),E*
FDCB d DC	SET 3,(IY+d),H*
FDCB d DD	SET 3,(IY+d),L*
FDCB d DE	SET 3,(IY+d)
FDCB d DF	SET 3,(IY+d),A*
FDCB d E0	SET 4,(IY+d),B*
FDCB d E1	SET 4,(IY+d),C*
FDCB d E2	SET 4,(IY+d),D*
FDCB d E3	SET 4,(IY+d),E*
FDCB d E4	SET 4,(IY+d),H*
FDCB d E5	SET 4,(IY+d),L*
FDCB d E6	SET 4,(IY+d)
FDCB d E7	SET 4,(IY+d),A*
FDCB d E8	SET 5,(IY+d),B*
FDCB d E9	SET 5,(IY+d),C*
FDCB d EA	SET 5,(IY+d),D*
FDCB d EB	SET 5,(IY+d),E*
FDCB d EC	SET 5,(IY+d),H*
FDCB d ED	SET 5,(IY+d),L*
FDCB d EE	SET 5,(IY+d)
FDCB d EF	SET 5,(IY+d),A*
FDCB d F0	SET 6,(IY+d),B*
FDCB d F1	SET 6,(IY+d),C*
FDCB d F2	SET 6,(IY+d),D*
FDCB d F3	SET 6,(IY+d),E*
FDCB d F4	SET 6,(IY+d),H*
FDCB d F5	SET 6,(IY+d),L*
FDCB d F6	SET 6,(IY+d)
FDCB d F7	SET 6,(IY+d),A*
FDCB d F8	SET 7,(IY+d),B*
FDCB d F9	SET 7,(IY+d),C*
FDCB d FA	SET 7,(IY+d),D*
FDCB d FB	SET 7,(IY+d),E*
FDCB d FC	SET 7,(IY+d),H*
FDCB d FD	SET 7,(IY+d),L*
FDCB d FE	SET 7,(IY+d)
FDCB d FF	SET 7,(IY+d),A*
FDE1	POP IY
FDE3	EX (SP),IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP,IY
FE n	CP n
FF	RST 38H

Chapter 10

Instructions Sorted by MNemonic

Any instruction marked with * is undocumented.

ADC A,(HL)	8E
ADC A,(IX+d)	DD8E d
ADC A,(IY+d)	FD8E d
ADC A,A	8F
ADC A,B	88
ADC A,C	89
ADC A,D	8A
ADC A,E	8B
ADC A,H	8C
ADC A,IXh*	DD8C
ADC A,IXl*	DD8D
ADC A,IYh*	FD8C
ADC A,IYl*	FD8D
ADC A,L	8D
ADC A,n	CE n
ADC HL,BC	ED4A
ADC HL,DE	ED5A
ADC HL,HL	ED6A
ADC HL,SP	ED7A
ADD A,(HL)	86
ADD A,(IX+d)	DD86 d
ADD A,(IY+d)	FD86 d
ADD A,A	87
ADD A,B	80
ADD A,C	81
ADD A,D	82
ADD A,E	83
ADD A,H	84
ADD A,IXh*	DD84
ADD A,IXl*	DD85
ADD A,IYh*	FD84
ADD A,IYl*	FD85
ADD A,L	85
ADD A,n	C6 n
ADD HL,BC	09
ADD HL,DE	19
ADD HL,HL	29
ADD HL,SP	39
ADD IX,BC	DD09
ADD IX,DE	DD19
ADD IX,IX	DD29
ADD IX,SP	DD39
ADD IY,BC	FD09
ADD IY,DE	FD19
ADD IY,IY	FD29
ADD IY,SP	FD39
AND (HL)	A6
AND (IX+d)	DDA6 d
AND (IY+d)	FDA6 d
AND A	A7
AND B	A0
AND C	A1
AND D	A2
AND E	A3
AND H	A4
AND IXh*	DDA4
AND IXl*	DDA5
AND IYh*	FDA4
AND IYl*	FDA5
AND L	A5
AND n	E6 n
BIT 0,(HL)	CB46
BIT 0,(IX+d)*	DDCB d 40
BIT 0,(IX+d)*	DDCB d 41
BIT 0,(IX+d)*	DDCB d 42
BIT 0,(IX+d)*	DDCB d 43
BIT 0,(IX+d)*	DDCB d 44
BIT 0,(IX+d)*	DDCB d 45
BIT 0,(IX+d)*	DDCB d 47
BIT 0,(IX+d)	DDCB d 46
BIT 0,(IY+d)*	FDCB d 40
BIT 0,(IY+d)*	FDCB d 41
BIT 0,(IY+d)*	FDCB d 42
BIT 0,(IY+d)*	FDCB d 43
BIT 0,(IY+d)*	FDCB d 44
BIT 0,(IY+d)*	FDCB d 45
BIT 0,(IY+d)*	FDCB d 47
BIT 0,(IY+d)	FDCB d 46
BIT 0,A	CB47
BIT 0,B	CB40
BIT 0,C	CB41
BIT 0,D	CB42
BIT 0,E	CB43
BIT 0,H	CB44
BIT 0,L	CB45
BIT 1,(HL)	CB4E
BIT 1,(IX+d)*	DDCB d 48
BIT 1,(IX+d)*	DDCB d 49
BIT 1,(IX+d)*	DDCB d 4A
BIT 1,(IX+d)*	DDCB d 4B
BIT 1,(IX+d)*	DDCB d 4C
BIT 1,(IX+d)*	DDCB d 4D
BIT 1,(IX+d)*	DDCB d 4F
BIT 1,(IX+d)	DDCB d 4E
BIT 1,(IY+d)*	FDCB d 48
BIT 1,(IY+d)*	FDCB d 49
BIT 1,(IY+d)*	FDCB d 4A
BIT 1,(IY+d)*	FDCB d 4B
BIT 1,(IY+d)*	FDCB d 4C
BIT 1,(IY+d)*	FDCB d 4D
BIT 1,(IY+d)*	FDCB d 4F
BIT 1,(IY+d)	FDCB d 4E
BIT 1,A	CB4F
BIT 1,B	CB48
BIT 1,C	CB49
BIT 1,D	CB4A
BIT 1,E	CB4B
BIT 1,H	CB4C
BIT 1,L	CB4D
BIT 2,(HL)	CB56
BIT 2,(IX+d)*	DDCB d 50
BIT 2,(IX+d)*	DDCB d 51
BIT 2,(IX+d)*	DDCB d 52
BIT 2,(IX+d)*	DDCB d 53
BIT 2,(IX+d)*	DDCB d 54
BIT 2,(IX+d)*	DDCB d 55
BIT 2,(IX+d)*	DDCB d 57
BIT 2,(IX+d)	DDCB d 56
BIT 2,(IY+d)*	FDCB d 50
BIT 2,(IY+d)*	FDCB d 51
BIT 2,(IY+d)*	FDCB d 52
BIT 2,(IY+d)*	FDCB d 53
BIT 2,(IY+d)*	FDCB d 54
BIT 2,(IY+d)*	FDCB d 55
BIT 2,(IY+d)*	FDCB d 57
BIT 2,(IY+d)	FDCB d 56
BIT 2,A	CB57
BIT 2,B	CB50
BIT 2,C	CB51
BIT 2,D	CB52
BIT 2,E	CB53
BIT 2,H	CB54
BIT 2,L	CB55
BIT 3,(HL)	CB5E
BIT 3,(IX+d)*	DDCB d 58
BIT 3,(IX+d)*	DDCB d 59
BIT 3,(IX+d)*	DDCB d 5A
BIT 3,(IX+d)*	DDCB d 5B
BIT 3,(IX+d)*	DDCB d 5C
BIT 3,(IX+d)*	DDCB d 5D
BIT 3,(IX+d)*	DDCB d 5F
BIT 3,(IX+d)	DDCB d 5E
BIT 3,(IY+d)*	FDCB d 58
BIT 3,(IY+d)*	FDCB d 59
BIT 3,(IY+d)*	FDCB d 5A
BIT 3,(IY+d)*	FDCB d 5B
BIT 3,(IY+d)*	FDCB d 5C
BIT 3,(IY+d)*	FDCB d 5D
BIT 3,(IY+d)*	FDCB d 5F
BIT 3,(IY+d)	FDCB d 5E
BIT 3,A	CB5F
BIT 3,B	CB58
BIT 3,C	CB59
BIT 3,D	CB5A
BIT 3,E	CB5B
BIT 3,H	CB5C
BIT 3,L	CB5D
BIT 4,(HL)	CB66
BIT 4,(IX+d)*	DDCB d 60
BIT 4,(IX+d)*	DDCB d 61
BIT 4,(IX+d)*	DDCB d 62
BIT 4,(IX+d)*	DDCB d 63
BIT 4,(IX+d)*	DDCB d 64
BIT 4,(IX+d)*	DDCB d 65
BIT 4,(IX+d)*	DDCB d 67

BIT 4,(IX+d)	DDCB d 66	BIT 7,(IY+d)	FDCB d 7E	IN C,(C)	ED48
BIT 4,(IY+d)*	FDCB d 60	BIT 7,A	CB7F	IN D,(C)	ED50
BIT 4,(IY+d)*	FDCB d 61	BIT 7,B	CB78	IN E,(C)	ED58
BIT 4,(IY+d)*	FDCB d 62	BIT 7,C	CB79	IN F,(C)* / IN (C)*	ED70
BIT 4,(IY+d)*	FDCB d 63	BIT 7,D	CB7A	IN H,(C)	ED60
BIT 4,(IY+d)*	FDCB d 64	BIT 7,E	CB7B	IN L,(C)	ED68
BIT 4,(IY+d)*	FDCB d 65	BIT 7,H	CB7C	INC (HL)	34
BIT 4,(IY+d)*	FDCB d 67	BIT 7,L	CB7D	INC (IX+d)	DD34 d
BIT 4,(IY+d)	FDCB d 66	CALL nn	CD n n	INC (IY+d)	FD34 d
BIT 4,A	CB67	CALL C,nn	DC n n	INC A	3C
BIT 4,B	CB60	CALL M,nn	FC n n	INC BC	03
BIT 4,C	CB61	CALL NC,nn	D4 n n	INC B	04
BIT 4,D	CB62	CALL NZ,nn	C4 n n	INC C	0C
BIT 4,E	CB63	CALL P,nn	F4 n n	INC DE	13
BIT 4,H	CB64	CALL PE,nn	EC n n	INC D	14
BIT 4,L	CB65	CALL PO,nn	E4 n n	INC E	1C
BIT 5,(HL)	CB6E	CALL Z,nn	CC n n	INC HL	23
BIT 5,(IX+d)*	DDCB d 68	CCF	3F	INC H	24
BIT 5,(IX+d)*	DDCB d 69	CP (HL)	BE	INC IX	DD23
BIT 5,(IX+d)*	DDCB d 6A	CP (IX+d)	DDBE d	INC IXh*	DD24
BIT 5,(IX+d)*	DDCB d 6B	CP (IY+d)	FDBE d	INC IXl*	DD2C
BIT 5,(IX+d)*	DDCB d 6C	CP A	BF	INC IY	FD23
BIT 5,(IX+d)*	DDCB d 6D	CP B	B8	INC IYh*	FD24
BIT 5,(IX+d)*	DDCB d 6F	CP C	B9	INC IYl*	FD2C
BIT 5,(IX+d)	DDCB d 6E	CP D	BA	INC L	2C
BIT 5,(IY+d)*	FDCB d 68	CP E	BB	INC SP	33
BIT 5,(IY+d)*	FDCB d 69	CP H	BC	INDR	EDBA
BIT 5,(IY+d)*	FDCB d 6A	CP IXh*	DDBC	IND	EDAA
BIT 5,(IY+d)*	FDCB d 6B	CP IXl*	DDBD	INIR	EDB2
BIT 5,(IY+d)*	FDCB d 6C	CP IYh*	FDBC	INI	EDA2
BIT 5,(IY+d)*	FDCB d 6D	CP IYl*	FDBD	JP (HL)	E9
BIT 5,(IY+d)*	FDCB d 6F	CP L	BD	JP (IX)	DDE9
BIT 5,(IY+d)	FDCB d 6E	CP n	FE n	JP (IY)	FDE9
BIT 5,A	CB6F	CPDR	EDB9	JP nn	C3 n n
BIT 5,B	CB68	CPD	EDA9	JP C,nn	DA n n
BIT 5,C	CB69	CPIR	EDB1	JP M,nn	FA n n
BIT 5,D	CB6A	CPI	EDA1	JP NC,nn	D2 n n
BIT 5,E	CB6B	CPL	2F	JP NZ,nn	C2 n n
BIT 5,H	CB6C	DAA	27	JP P,nn	F2 n n
BIT 5,L	CB6D	DEC (HL)	35	JP PE,nn	EA n n
BIT 6,(HL)	CB76	DEC (IX+d)	DD35 d	JP PO,nn	E2 n n
BIT 6,(IX+d)*	DDCB d 70	DEC (IY+d)	FD35 d	JP Z,nn	CA n n
BIT 6,(IX+d)*	DDCB d 71	DEC A	3D	JR e	18 e
BIT 6,(IX+d)*	DDCB d 72	DEC BC	0B	JR C,e	38 e
BIT 6,(IX+d)*	DDCB d 73	DEC B	05	JR NC,e	30 e
BIT 6,(IX+d)*	DDCB d 74	DEC C	0D	JR NZ,e	20 e
BIT 6,(IX+d)*	DDCB d 75	DEC DE	1B	JR Z,e	28 e
BIT 6,(IX+d)*	DDCB d 77	DEC D	15	LD (BC),A	02
BIT 6,(IX+d)	DDCB d 76	DEC E	1D	LD (DE),A	12
BIT 6,(IY+d)*	FDCB d 70	DEC HL	2B	LD (HL),A	77
BIT 6,(IY+d)*	FDCB d 71	DEC H	25	LD (HL),B	70
BIT 6,(IY+d)*	FDCB d 72	DEC IX	DD2B	LD (HL),C	71
BIT 6,(IY+d)*	FDCB d 73	DEC IXh*	DD25	LD (HL),D	72
BIT 6,(IY+d)*	FDCB d 74	DEC IXl*	DD2D	LD (HL),E	73
BIT 6,(IY+d)*	FDCB d 75	DEC IY	FD2B	LD (HL),H	74
BIT 6,(IY+d)*	FDCB d 77	DEC IYh*	FD25	LD (HL),L	75
BIT 6,(IY+d)	FDCB d 76	DEC IYl*	FD2D	LD (HL),n	36 n
BIT 6,A	CB77	DEC L	2D	LD (IX+d),A	DD77 d
BIT 6,B	CB70	DEC SP	3B	LD (IX+d),B	DD70 d
BIT 6,C	CB71	DI	F3	LD (IX+d),C	DD71 d
BIT 6,D	CB72	DJNZ (PC+e)	10 e	LD (IX+d),D	DD72 d
BIT 6,E	CB73	EI	FB	LD (IX+d),E	DD73 d
BIT 6,H	CB74	EX (SP),HL	E3	LD (IX+d),H	DD74 d
BIT 6,L	CB75	EX (SP),IX	DDE3	LD (IX+d),L	DD75 d
BIT 7,(HL)	CB7E	EX (SP),IY	FDE3	LD (IX+d),n	DD36 d n
BIT 7,(IX+d)*	DDCB d 78	EX AF,AF'	08	LD (IY+d),A	FD77 d
BIT 7,(IX+d)*	DDCB d 79	EX DE,HL	EB	LD (IY+d),B	FD70 d
BIT 7,(IX+d)*	DDCB d 7A	EXX	D9	LD (IY+d),C	FD71 d
BIT 7,(IX+d)*	DDCB d 7B	HALT	76	LD (IY+d),D	FD72 d
BIT 7,(IX+d)*	DDCB d 7C	IM 0*	ED4E	LD (IY+d),E	FD73 d
BIT 7,(IX+d)*	DDCB d 7D	IM 0*	ED66	LD (IY+d),H	FD74 d
BIT 7,(IX+d)*	DDCB d 7F	IM 0*	ED6E	LD (IY+d),L	FD75 d
BIT 7,(IX+d)	DDCB d 7E	IM 0	ED46	LD (IY+d),n	FD36 d n
BIT 7,(IY+d)*	FDCB d 78	IM 1*	ED76	LD (nn),A	32 n n
BIT 7,(IY+d)*	FDCB d 79	IM 1	ED56	LD (nn),BC	ED43 n n
BIT 7,(IY+d)*	FDCB d 7A	IM 2*	ED7E	LD (nn),DE	ED53 n n
BIT 7,(IY+d)*	FDCB d 7B	IM 2	ED5E	LD (nn),HL	22 n n
BIT 7,(IY+d)*	FDCB d 7C	IN A,(C)	ED78	LD (nn),HL	ED63 n n
BIT 7,(IY+d)*	FDCB d 7D	IN A,(n)	DB n	LD (nn),IX	DD22 n n
BIT 7,(IY+d)*	FDCB d 7F	IN B,(C)	ED40	LD (nn),IY	FD22 n n

LD (nn),SP	ED73 n n
LD A,(BC)	0A
LD A,(DE)	1A
LD A,(HL)	7E
LD A,(IX+d)	DD7E d
LD A,(IY+d)	FD7E d
LD A,(nn)	3A n n
LD A,A	7F
LD A,B	78
LD A,C	79
LD A,D	7A
LD A,E	7B
LD A,H	7C
LD A,IXh*	DD7C
LD A,IXl*	DD7D
LD A,IYh*	FD7C
LD A,IYl*	FD7D
LD A,I	ED57
LD A,L	7D
LD A,R	ED5F
LD A,n	3E n
LD B,(HL)	46
LD B,(IX+d)	DD46 d
LD B,(IY+d)	FD46 d
LD B,A	47
LD B,B	40
LD B,C	41
LD B,D	42
LD B,E	43
LD B,H	44
LD B,IXh*	DD44
LD B,IXl*	DD45
LD B,IYh*	FD44
LD B,IYl*	FD45
LD B,L	45
LD B,n	06 n
LD BC,(nn)	ED4B n n
LD BC,nn	01 n n
LD C,(HL)	4E
LD C,(IX+d)	DD4E d
LD C,(IY+d)	FD4E d
LD C,A	4F
LD C,B	48
LD C,C	49
LD C,D	4A
LD C,E	4B
LD C,H	4C
LD C,IXh*	DD4C
LD C,IXl*	DD4D
LD C,IYh*	FD4C
LD C,IYl*	FD4D
LD C,L	4D
LD C,n	0E n
LD D,(HL)	56
LD D,(IX+d)	DD56 d
LD D,(IY+d)	FD56 d
LD D,A	57
LD D,B	50
LD D,C	51
LD D,D	52
LD D,E	53
LD D,H	54
LD D,IXh*	DD54
LD D,IXl*	DD55
LD D,IYh*	FD54
LD D,IYl*	FD55
LD D,L	55
LD D,n	16 n
LD DE,(nn)	ED5B n n
LD DE,nn	11 n n
LD E,(HL)	5E
LD E,(IX+d)	DD5E d
LD E,(IY+d)	FD5E d
LD E,A	5F
LD E,B	58
LD E,C	59
LD E,D	5A
LD E,E	5B
LD E,H	5C
LD E,IXh*	DD5C
LD E,IXl*	DD5D
LD E,IYh*	FD5C
LD E,IYl*	FD5D
LD E,L	5D
LD E,n	1E n
LD H,(HL)	66
LD H,(IX+d)	DD66 d
LD H,(IY+d)	FD66 d
LD H,A	67
LD H,B	60
LD H,C	61
LD H,D	62
LD H,E	63
LD H,H	64
LD H,L	65
LD H,n	26 n
LD HL,(nn)	2A n n
LD HL,(nn)	ED6B n n
LD HL,nn	21 n n
LD I,A	ED47
LD IX,(nn)	DD2A n n
LD IX,nn	DD21 n n
LD IXh,A*	DD67
LD IXh,B*	DD60
LD IXh,C*	DD61
LD IXh,D*	DD62
LD IXh,E*	DD63
LD IXh,IXh*	DD64
LD IXh,IXl*	DD65
LD IXh,n*	DD26 n
LD IXl,A*	DD6F
LD IXl,B*	DD68
LD IXl,C*	DD69
LD IXl,D*	DD6A
LD IXl,E*	DD6B
LD IXl,IXh*	DD6C
LD IXl,IXl*	DD6D
LD IXl,n*	DD2E n
LD IY,(nn)	FD2A n n
LD IY,nn	FD21 n n
LD IYh,A*	FD67
LD IYh,B*	FD60
LD IYh,C*	FD61
LD IYh,D*	FD62
LD IYh,E*	FD63
LD IYh,IYh*	FD64
LD IYh,IYl*	FD65
LD IYh,n*	FD26 n
LD IYl,A*	FD6F
LD IYl,B*	FD68
LD IYl,C*	FD69
LD IYl,D*	FD6A
LD IYl,E*	FD6B
LD IYl,IYh*	FD6C
LD IYl,IYl*	FD6D
LD IYl,n*	FD2E n
LD L,(HL)	6E
LD L,(IX+d)	DD6E d
LD L,(IY+d)	FD6E d
LD L,A	6F
LD L,B	68
LD L,C	69
LD L,D	6A
LD L,E	6B
LD L,H	6C
LD L,L	6D
LD L,n	2E n
LD R,A	ED4F
LD SP,(nn)	ED7B n n
LD SP,HL	F9
LD SP,IX	DDF9
LD SP,IY	FDF9
LD SP,nn	31 n n
LDDR	EDB8
LDD	EDA8
LDIR	EDB0
LDI	EDA0
NEG*	ED4C
NEG*	ED54
NEG*	ED5C
NEG*	ED64
NEG*	ED6C
NEG*	ED74
NEG*	ED7C
NEG	ED44
NOP	00
OR (HL)	B6
OR (IX+d)	DDB6 d
OR (IY+d)	FDB6 d
OR A	B7
OR B	B0
OR C	B1
OR D	B2
OR E	B3
OR H	B4
OR IXh*	DDB4
OR IXl*	DDB5
OR IYh*	FDB4
OR IYl*	FDB5
OR L	B5
OR n	F6 n
OTDR	EDBB
OTIR	EDB3
OUT (C),0*	ED71
OUT (C),A	ED79
OUT (C),B	ED41
OUT (C),C	ED49
OUT (C),D	ED51
OUT (C),E	ED59
OUT (C),H	ED61
OUT (C),L	ED69
OUT (n),A	D3 n
OUTD	EDAB
OUTI	EDA3
POP AF	F1
POP BC	C1
POP DE	D1
POP HL	E1
POP IX	DDE1
POP IY	FDE1
PUSH AF	F5
PUSH BC	C5
PUSH DE	D5
PUSH HL	E5
PUSH IX	DDE5
PUSH IY	FDE5
RES 0,(HL)	CB86
RES 0,(IX+d),A*	DDCB d 87
RES 0,(IX+d),B*	DDCB d 80
RES 0,(IX+d),C*	DDCB d 81
RES 0,(IX+d),D*	DDCB d 82
RES 0,(IX+d),E*	DDCB d 83
RES 0,(IX+d),H*	DDCB d 84
RES 0,(IX+d),L*	DDCB d 85
RES 0,(IX+d)	DDCB d 86
RES 0,(IY+d),A*	FDCB d 87
RES 0,(IY+d),B*	FDCB d 80
RES 0,(IY+d),C*	FDCB d 81
RES 0,(IY+d),D*	FDCB d 82
RES 0,(IY+d),E*	FDCB d 83
RES 0,(IY+d),H*	FDCB d 84
RES 0,(IY+d),L*	FDCB d 85
RES 0,(IY+d)	FDCB d 86
RES 0,A	CB87
RES 0,B	CB80
RES 0,C	CB81
RES 0,D	CB82
RES 0,E	CB83
RES 0,H	CB84
RES 0,L	CB85
RES 1,(HL)	CB8E
RES 1,(IX+d),A*	DDCB d 8F
RES 1,(IX+d),B*	DDCB d 88
RES 1,(IX+d),C*	DDCB d 89
RES 1,(IX+d),D*	DDCB d 8A
RES 1,(IX+d),E*	DDCB d 8B
RES 1,(IX+d),H*	DDCB d 8C
RES 1,(IX+d),L*	DDCB d 8D
RES 1,(IX+d)	DDCB d 8E
RES 1,(IY+d),A*	FDCB d 8F

RES 1,(IY+d),B*	FDCB d 88	RES 4,B	CBA0	RET NC	D0
RES 1,(IY+d),C*	FDCB d 89	RES 4,C	CBA1	RET NZ	C0
RES 1,(IY+d),D*	FDCB d 8A	RES 4,D	CBA2	RET PE	E8
RES 1,(IY+d),E*	FDCB d 8B	RES 4,E	CBA3	RET PO	E0
RES 1,(IY+d),H*	FDCB d 8C	RES 4,H	CBA4	RET P	F0
RES 1,(IY+d),L*	FDCB d 8D	RES 4,L	CBA5	RET Z	C8
RES 1,(IY+d)	FDCB d 8E	RES 5,(HL)	CBAE	RETI	ED4D
RES 1,A	CB8F	RES 5,(IX+d),A*	DDCB d AF	RETN*	ED55
RES 1,B	CB88	RES 5,(IX+d),B*	DDCB d A8	RETN*	ED5D
RES 1,C	CB89	RES 5,(IX+d),C*	DDCB d A9	RETN*	ED65
RES 1,D	CB8A	RES 5,(IX+d),D*	DDCB d AA	RETN*	ED6D
RES 1,E	CB8B	RES 5,(IX+d),E*	DDCB d AB	RETN*	ED75
RES 1,H	CB8C	RES 5,(IX+d),H*	DDCB d AC	RETN*	ED7D
RES 1,L	CB8D	RES 5,(IX+d),L*	DDCB d AD	RETN	ED45
RES 2,(HL)	CB96	RES 5,(IX+d)	DDCB d AE	RET	C9
RES 2,(IX+d),A*	DDCB d 97	RES 5,(IY+d),A*	FDCB d AF	RL (HL)	CB16
RES 2,(IX+d),B*	DDCB d 90	RES 5,(IY+d),B*	FDCB d A8	RL (IX+d),A*	DDCB d 17
RES 2,(IX+d),C*	DDCB d 91	RES 5,(IY+d),C*	FDCB d A9	RL (IX+d),B*	DDCB d 10
RES 2,(IX+d),D*	DDCB d 92	RES 5,(IY+d),D*	FDCB d AA	RL (IX+d),C*	DDCB d 11
RES 2,(IX+d),E*	DDCB d 93	RES 5,(IY+d),E*	FDCB d AB	RL (IX+d),D*	DDCB d 12
RES 2,(IX+d),H*	DDCB d 94	RES 5,(IY+d),H*	FDCB d AC	RL (IX+d),E*	DDCB d 13
RES 2,(IX+d),L*	DDCB d 95	RES 5,(IY+d),L*	FDCB d AD	RL (IX+d),H*	DDCB d 14
RES 2,(IX+d)	DDCB d 96	RES 5,(IY+d)	FDCB d AE	RL (IX+d),L*	DDCB d 15
RES 2,(IY+d),A*	FDCB d 97	RES 5,A	CBAF	RL (IX+d)	DDCB d 16
RES 2,(IY+d),B*	FDCB d 90	RES 5,B	CBA8	RL (IY+d),A*	FDCB d 17
RES 2,(IY+d),C*	FDCB d 91	RES 5,C	CBA9	RL (IY+d),B*	FDCB d 10
RES 2,(IY+d),D*	FDCB d 92	RES 5,D	CBAA	RL (IY+d),C*	FDCB d 11
RES 2,(IY+d),E*	FDCB d 93	RES 5,E	CBAB	RL (IY+d),D*	FDCB d 12
RES 2,(IY+d),H*	FDCB d 94	RES 5,H	CBAC	RL (IY+d),E*	FDCB d 13
RES 2,(IY+d),L*	FDCB d 95	RES 5,L	CBAD	RL (IY+d),H*	FDCB d 14
RES 2,(IY+d)	FDCB d 96	RES 6,(HL)	CBB6	RL (IY+d),L*	FDCB d 15
RES 2,A	CB97	RES 6,(IX+d),A*	DDCB d B7	RL (IY+d)	FDCB d 16
RES 2,B	CB90	RES 6,(IX+d),B*	DDCB d B0	RL A	CB17
RES 2,C	CB91	RES 6,(IX+d),C*	DDCB d B1	RL B	CB10
RES 2,D	CB92	RES 6,(IX+d),D*	DDCB d B2	RL C	CB11
RES 2,E	CB93	RES 6,(IX+d),E*	DDCB d B3	RL D	CB12
RES 2,H	CB94	RES 6,(IX+d),H*	DDCB d B4	RL E	CB13
RES 2,L	CB95	RES 6,(IX+d),L*	DDCB d B5	RL H	CB14
RES 3,(HL)	CB9E	RES 6,(IX+d)	DDCB d B6	RL L	CB15
RES 3,(IX+d),A*	DDCB d 9F	RES 6,(IY+d),A*	FDCB d B7	RLA	17
RES 3,(IX+d),B*	DDCB d 98	RES 6,(IY+d),B*	FDCB d B0	RLC (HL)	CB06
RES 3,(IX+d),C*	DDCB d 99	RES 6,(IY+d),C*	FDCB d B1	RLC (IX+d),A*	DDCB d 07
RES 3,(IX+d),D*	DDCB d 9A	RES 6,(IY+d),D*	FDCB d B2	RLC (IX+d),B*	DDCB d 00
RES 3,(IX+d),E*	DDCB d 9B	RES 6,(IY+d),E*	FDCB d B3	RLC (IX+d),C*	DDCB d 01
RES 3,(IX+d),H*	DDCB d 9C	RES 6,(IY+d),H*	FDCB d B4	RLC (IX+d),D*	DDCB d 02
RES 3,(IX+d),L*	DDCB d 9D	RES 6,(IY+d),L*	FDCB d B5	RLC (IX+d),E*	DDCB d 03
RES 3,(IX+d)	DDCB d 9E	RES 6,(IY+d)	FDCB d B6	RLC (IX+d),H*	DDCB d 04
RES 3,(IY+d),A*	FDCB d 9F	RES 6,A	CBB7	RLC (IX+d),L*	DDCB d 05
RES 3,(IY+d),B*	FDCB d 98	RES 6,B	CBB0	RLC (IX+d)	DDCB d 06
RES 3,(IY+d),C*	FDCB d 99	RES 6,C	CBB1	RLC (IY+d),A*	FDCB d 07
RES 3,(IY+d),D*	FDCB d 9A	RES 6,D	CBB2	RLC (IY+d),B*	FDCB d 00
RES 3,(IY+d),E*	FDCB d 9B	RES 6,E	CBB3	RLC (IY+d),C*	FDCB d 01
RES 3,(IY+d),H*	FDCB d 9C	RES 6,H	CBB4	RLC (IY+d),D*	FDCB d 02
RES 3,(IY+d),L*	FDCB d 9D	RES 6,L	CBB5	RLC (IY+d),E*	FDCB d 03
RES 3,(IY+d)	FDCB d 9E	RES 7,(HL)	CBBE	RLC (IY+d),H*	FDCB d 04
RES 3,A	CB9F	RES 7,(IX+d),A*	DDCB d BF	RLC (IY+d),L*	FDCB d 05
RES 3,B	CB98	RES 7,(IX+d),B*	DDCB d B8	RLC (IY+d)	FDCB d 06
RES 3,C	CB99	RES 7,(IX+d),C*	DDCB d B9	RLC A	CB07
RES 3,D	CB9A	RES 7,(IX+d),D*	DDCB d BA	RLC B	CB00
RES 3,E	CB9B	RES 7,(IX+d),E*	DDCB d BB	RLC C	CB01
RES 3,H	CB9C	RES 7,(IX+d),H*	DDCB d BC	RLC D	CB02
RES 3,L	CB9D	RES 7,(IX+d),L*	DDCB d BD	RLC E	CB03
RES 4,(HL)	CBA6	RES 7,(IX+d)	DDCB d BE	RLC H	CB04
RES 4,(IX+d),A*	DDCB d A7	RES 7,(IY+d),A*	FDCB d BF	RLC L	CB05
RES 4,(IX+d),B*	DDCB d A0	RES 7,(IY+d),B*	FDCB d B8	RLCA	07
RES 4,(IX+d),C*	DDCB d A1	RES 7,(IY+d),C*	FDCB d B9	RLD	ED6F
RES 4,(IX+d),D*	DDCB d A2	RES 7,(IY+d),D*	FDCB d BA	RR (HL)	CB1E
RES 4,(IX+d),E*	DDCB d A3	RES 7,(IY+d),E*	FDCB d BB	RR (IX+d),A*	DDCB d 1F
RES 4,(IX+d),H*	DDCB d A4	RES 7,(IY+d),H*	FDCB d BC	RR (IX+d),B*	DDCB d 18
RES 4,(IX+d),L*	DDCB d A5	RES 7,(IY+d),L*	FDCB d BD	RR (IX+d),C*	DDCB d 19
RES 4,(IX+d)	DDCB d A6	RES 7,(IY+d)	FDCB d BE	RR (IX+d),D*	DDCB d 1A
RES 4,(IY+d),A*	FDCB d A7	RES 7,A	CBBF	RR (IX+d),E*	DDCB d 1B
RES 4,(IY+d),B*	FDCB d A0	RES 7,B	CBB8	RR (IX+d),H*	DDCB d 1C
RES 4,(IY+d),C*	FDCB d A1	RES 7,C	CBB9	RR (IX+d),L*	DDCB d 1D
RES 4,(IY+d),D*	FDCB d A2	RES 7,D	CBBA	RR (IX+d)	DDCB d 1E
RES 4,(IY+d),E*	FDCB d A3	RES 7,E	CBBB	RR (IY+d),A*	FDCB d 1F
RES 4,(IY+d),H*	FDCB d A4	RES 7,H	CBBC	RR (IY+d),B*	FDCB d 18
RES 4,(IY+d),L*	FDCB d A5	RES 7,L	CBBD	RR (IY+d),C*	FDCB d 19
RES 4,(IY+d)	FDCB d A6	RET C	D8	RR (IY+d),D*	FDCB d 1A
RES 4,A	CBA7	RET M	F8	RR (IY+d),E*	FDCB d 1B

RR (IY+d),H*	FDCB d 1C	SET 0,(IY+d),L*	FDCB d C5	SET 3,L	CBDD
RR (IY+d),L*	FDCB d 1D	SET 0,(IY+d)	FDCB d C6	SET 4,(HL)	CBE6
RR (IY+d)	FDCB d 1E	SET 0,A	CBC7	SET 4,(IX+d),A*	DDCB d E7
RR A	CB1F	SET 0,B	CBC0	SET 4,(IX+d),B*	DDCB d E0
RR B	CB18	SET 0,C	CBC1	SET 4,(IX+d),C*	DDCB d E1
RR C	CB19	SET 0,D	CBC2	SET 4,(IX+d),D*	DDCB d E2
RR D	CB1A	SET 0,E	CBC3	SET 4,(IX+d),E*	DDCB d E3
RR E	CB1B	SET 0,H	CBC4	SET 4,(IX+d),H*	DDCB d E4
RR H	CB1C	SET 0,L	CBC5	SET 4,(IX+d),L*	DDCB d E5
RR L	CB1D	SET 1,(HL)	CBCE	SET 4,(IX+d)	DDCB d E6
RRA	1F	SET 1,(IX+d),A*	DDCB d CF	SET 4,(IY+d),A*	FDCB d E7
RRC (HL)	CB0E	SET 1,(IX+d),B*	DDCB d C8	SET 4,(IY+d),B*	FDCB d E0
RRC (IX+d),A*	DDCB d 0F	SET 1,(IX+d),C*	DDCB d C9	SET 4,(IY+d),C*	FDCB d E1
RRC (IX+d),B*	DDCB d 08	SET 1,(IX+d),D*	DDCB d CA	SET 4,(IY+d),D*	FDCB d E2
RRC (IX+d),C*	DDCB d 09	SET 1,(IX+d),E*	DDCB d CB	SET 4,(IY+d),E*	FDCB d E3
RRC (IX+d),D*	DDCB d 0A	SET 1,(IX+d),H*	DDCB d CC	SET 4,(IY+d),H*	FDCB d E4
RRC (IX+d),E*	DDCB d 0B	SET 1,(IX+d),L*	DDCB d CD	SET 4,(IY+d),L*	FDCB d E5
RRC (IX+d),H*	DDCB d 0C	SET 1,(IX+d)	DDCB d CE	SET 4,(IY+d)	FDCB d E6
RRC (IX+d),L*	DDCB d 0D	SET 1,(IY+d),A*	FDCB d CF	SET 4,A	CBE7
RRC (IX+d)	DDCB d 0E	SET 1,(IY+d),B*	FDCB d C8	SET 4,B	CBE0
RRC (IY+d),A*	FDCB d 0F	SET 1,(IY+d),C*	FDCB d C9	SET 4,C	CBE1
RRC (IY+d),B*	FDCB d 08	SET 1,(IY+d),D*	FDCB d CA	SET 4,D	CBE2
RRC (IY+d),C*	FDCB d 09	SET 1,(IY+d),E*	FDCB d CB	SET 4,E	CBE3
RRC (IY+d),D*	FDCB d 0A	SET 1,(IY+d),H*	FDCB d CC	SET 4,H	CBE4
RRC (IY+d),E*	FDCB d 0B	SET 1,(IY+d),L*	FDCB d CD	SET 4,L	CBE5
RRC (IY+d),H*	FDCB d 0C	SET 1,(IY+d)	FDCB d CE	SET 5,(HL)	CBEE
RRC (IY+d),L*	FDCB d 0D	SET 1,A	CBCF	SET 5,(IX+d),A*	DDCB d EF
RRC (IY+d)	FDCB d 0E	SET 1,B	CBC8	SET 5,(IX+d),B*	DDCB d E8
RRC A	CB0F	SET 1,C	CBC9	SET 5,(IX+d),C*	DDCB d E9
RRC B	CB08	SET 1,D	CBCA	SET 5,(IX+d),D*	DDCB d EA
RRC C	CB09	SET 1,E	CBCB	SET 5,(IX+d),E*	DDCB d EB
RRC D	CB0A	SET 1,H	CBCC	SET 5,(IX+d),H*	DDCB d EC
RRC E	CB0B	SET 1,L	CBCD	SET 5,(IX+d),L*	DDCB d ED
RRC H	CB0C	SET 2,(HL)	CBD6	SET 5,(IX+d)	DDCB d EE
RRC L	CB0D	SET 2,(IX+d),A*	DDCB d D7	SET 5,(IY+d),A*	FDCB d EF
RRCA	0F	SET 2,(IX+d),B*	DDCB d D0	SET 5,(IY+d),B*	FDCB d E8
RRD	ED67	SET 2,(IX+d),C*	DDCB d D1	SET 5,(IY+d),C*	FDCB d E9
RST 0H	C7	SET 2,(IX+d),D*	DDCB d D2	SET 5,(IY+d),D*	FDCB d EA
RST 10H	D7	SET 2,(IX+d),E*	DDCB d D3	SET 5,(IY+d),E*	FDCB d EB
RST 18H	DF	SET 2,(IX+d),H*	DDCB d D4	SET 5,(IY+d),H*	FDCB d EC
RST 20H	E7	SET 2,(IX+d),L*	DDCB d D5	SET 5,(IY+d),L*	FDCB d ED
RST 28H	EF	SET 2,(IX+d)	DDCB d D6	SET 5,(IY+d)	FDCB d EE
RST 30H	F7	SET 2,(IY+d),A*	FDCB d D7	SET 5,A	CBEF
RST 38H	FF	SET 2,(IY+d),B*	FDCB d D0	SET 5,B	CBE8
RST 8H	CF	SET 2,(IY+d),C*	FDCB d D1	SET 5,C	CBE9
SBC A,(HL)	9E	SET 2,(IY+d),D*	FDCB d D2	SET 5,D	CBEA
SBC A,(IX+d)	DD9E d	SET 2,(IY+d),E*	FDCB d D3	SET 5,E	CBEB
SBC A,(IY+d)	FD9E d	SET 2,(IY+d),H*	FDCB d D4	SET 5,H	CBEC
SBC A,A	9F	SET 2,(IY+d),L*	FDCB d D5	SET 5,L	CBED
SBC A,B	98	SET 2,(IY+d)	FDCB d D6	SET 6,(HL)	CBF6
SBC A,C	99	SET 2,A	CBD7	SET 6,(IX+d),A*	DDCB d F7
SBC A,D	9A	SET 2,B	CBD0	SET 6,(IX+d),B*	DDCB d F0
SBC A,E	9B	SET 2,C	CBD1	SET 6,(IX+d),C*	DDCB d F1
SBC A,H	9C	SET 2,D	CBD2	SET 6,(IX+d),D*	DDCB d F2
SBC A,IXh*	DD9C	SET 2,E	CBD3	SET 6,(IX+d),E*	DDCB d F3
SBC A,IXl*	DD9D	SET 2,H	CBD4	SET 6,(IX+d),H*	DDCB d F4
SBC A,IYh*	FD9C	SET 2,L	CBD5	SET 6,(IX+d),L*	DDCB d F5
SBC A,IYl*	FD9D	SET 3,(HL)	CBDE	SET 6,(IX+d)	DDCB d F6
SBC A,L	9D	SET 3,(IX+d),A*	DDCB d DF	SET 6,(IY+d),A*	FDCB d F7
SBC A,n	DE n	SET 3,(IX+d),B*	DDCB d D8	SET 6,(IY+d),B*	FDCB d F0
SBC HL,BC	ED42	SET 3,(IX+d),C*	DDCB d D9	SET 6,(IY+d),C*	FDCB d F1
SBC HL,DE	ED52	SET 3,(IX+d),D*	DDCB d DA	SET 6,(IY+d),D*	FDCB d F2
SBC HL,HL	ED62	SET 3,(IX+d),E*	DDCB d DB	SET 6,(IY+d),E*	FDCB d F3
SBC HL,SP	ED72	SET 3,(IX+d),H*	DDCB d DC	SET 6,(IY+d),H*	FDCB d F4
SCF	37	SET 3,(IX+d),L*	DDCB d DD	SET 6,(IY+d),L*	FDCB d F5
SET 0,(HL)	CBC6	SET 3,(IX+d)	DDCB d DE	SET 6,(IY+d)	FDCB d F6
SET 0,(IX+d),A*	DDCB d C7	SET 3,(IY+d),A*	FDCB d DF	SET 6,A	CBF7
SET 0,(IX+d),B*	DDCB d C0	SET 3,(IY+d),B*	FDCB d D8	SET 6,B	CBF0
SET 0,(IX+d),C*	DDCB d C1	SET 3,(IY+d),C*	FDCB d D9	SET 6,C	CBF1
SET 0,(IX+d),D*	DDCB d C2	SET 3,(IY+d),D*	FDCB d DA	SET 6,D	CBF2
SET 0,(IX+d),E*	DDCB d C3	SET 3,(IY+d),E*	FDCB d DB	SET 6,E	CBF3
SET 0,(IX+d),H*	DDCB d C4	SET 3,(IY+d),H*	FDCB d DC	SET 6,H	CBF4
SET 0,(IX+d),L*	DDCB d C5	SET 3,(IY+d),L*	FDCB d DD	SET 6,L	CBF5
SET 0,(IX+d)	DDCB d C6	SET 3,(IY+d)	FDCB d DE	SET 7,(HL)	CBFE
SET 0,(IY+d),A*	FDCB d C7	SET 3,A	CBDF	SET 7,(IX+d),A*	DDCB d FF
SET 0,(IY+d),B*	FDCB d C0	SET 3,B	CBD8	SET 7,(IX+d),B*	DDCB d F8
SET 0,(IY+d),C*	FDCB d C1	SET 3,C	CBD9	SET 7,(IX+d),C*	DDCB d F9
SET 0,(IY+d),D*	FDCB d C2	SET 3,D	CBDA	SET 7,(IX+d),D*	DDCB d FA
SET 0,(IY+d),E*	FDCB d C3	SET 3,E	CBDB	SET 7,(IX+d),E*	DDCB d FB
SET 0,(IY+d),H*	FDCB d C4	SET 3,H	CBDC	SET 7,(IX+d),H*	DDCB d FC

SET 7,(IX+d),L*	DDCB d FD
SET 7,(IX+d)	DDCB d FE
SET 7,(IY+d),A*	FDCB d FF
SET 7,(IY+d),B*	FDCB d F8
SET 7,(IY+d),C*	FDCB d F9
SET 7,(IY+d),D*	FDCB d FA
SET 7,(IY+d),E*	FDCB d FB
SET 7,(IY+d),H*	FDCB d FC
SET 7,(IY+d),L*	FDCB d FD
SET 7,(IY+d)	FDCB d FE
SET 7,A	CBFF
SET 7,B	CBF8
SET 7,C	CBF9
SET 7,D	CBFA
SET 7,E	CBFB
SET 7,H	CBFC
SET 7,L	CBFD
SLA (HL)	CB26
SLA (IX+d),A*	DDCB d 27
SLA (IX+d),B*	DDCB d 20
SLA (IX+d),C*	DDCB d 21
SLA (IX+d),D*	DDCB d 22
SLA (IX+d),E*	DDCB d 23
SLA (IX+d),H*	DDCB d 24
SLA (IX+d),L*	DDCB d 25
SLA (IX+d)	DDCB d 26
SLA (IY+d),A*	FDCB d 27
SLA (IY+d),B*	FDCB d 20
SLA (IY+d),C*	FDCB d 21
SLA (IY+d),D*	FDCB d 22
SLA (IY+d),E*	FDCB d 23
SLA (IY+d),H*	FDCB d 24
SLA (IY+d),L*	FDCB d 25
SLA (IY+d)	FDCB d 26
SLA A	CB27
SLA B	CB20
SLA C	CB21
SLA D	CB22
SLA E	CB23
SLA H	CB24
SLA L	CB25
SLL (HL)*	CB36
SLL (IX+d)*	DDCB d 36
SLL (IX+d),A*	DDCB d 37
SLL (IX+d),B*	DDCB d 30
SLL (IX+d),C*	DDCB d 31
SLL (IX+d),D*	DDCB d 32
SLL (IX+d),E*	DDCB d 33
SLL (IX+d),H*	DDCB d 34
SLL (IX+d),L*	DDCB d 35
SLL (IY+d)*	FDCB d 36
SLL (IY+d),A*	FDCB d 37
SLL (IY+d),B*	FDCB d 30
SLL (IY+d),C*	FDCB d 31
SLL (IY+d),D*	FDCB d 32
SLL (IY+d),E*	FDCB d 33
SLL (IY+d),H*	FDCB d 34
SLL (IY+d),L*	FDCB d 35
SLL A*	CB37
SLL B*	CB30
SLL C*	CB31
SLL D*	CB32
SLL E*	CB33
SLL H*	CB34
SLL L*	CB35
SRA (HL)	CB2E
SRA (IX+d),A*	DDCB d 2F
SRA (IX+d),B*	DDCB d 28
SRA (IX+d),C*	DDCB d 29
SRA (IX+d),D*	DDCB d 2A
SRA (IX+d),E*	DDCB d 2B
SRA (IX+d),H*	DDCB d 2C
SRA (IX+d),L*	DDCB d 2D
SRA (IX+d)	DDCB d 2E
SRA (IY+d),A*	FDCB d 2F
SRA (IY+d),B*	FDCB d 28
SRA (IY+d),C*	FDCB d 29
SRA (IY+d),D*	FDCB d 2A
SRA (IY+d),E*	FDCB d 2B
SRA (IY+d),H*	FDCB d 2C
SRA (IY+d),L*	FDCB d 2D
SRA (IY+d)	FDCB d 2E
SRA A	CB2F
SRA B	CB28
SRA C	CB29
SRA D	CB2A
SRA E	CB2B
SRA H	CB2C
SRA L	CB2D
SRL (HL)	CB3E
SRL (IX+d),A*	DDCB d 3F
SRL (IX+d),B*	DDCB d 38
SRL (IX+d),C*	DDCB d 39
SRL (IX+d),D*	DDCB d 3A
SRL (IX+d),E*	DDCB d 3B
SRL (IX+d),H*	DDCB d 3C
SRL (IX+d),L*	DDCB d 3D
SRL (IX+d)	DDCB d 3E
SRL (IY+d),A*	FDCB d 3F
SRL (IY+d),B*	FDCB d 38
SRL (IY+d),C*	FDCB d 39
SRL (IY+d),D*	FDCB d 3A
SRL (IY+d),E*	FDCB d 3B
SRL (IY+d),H*	FDCB d 3C
SRL (IY+d),L*	FDCB d 3D
SRL (IY+d)	FDCB d 3E
SRL A	CB3F
SRL B	CB38
SRL C	CB39
SRL D	CB3A
SRL E	CB3B
SRL H	CB3C
SRL L	CB3D
SUB (HL)	96
SUB (IX+d)	DD96 d
SUB (IY+d)	FD96 d
SUB A	97
SUB B	90
SUB C	91
SUB D	92
SUB E	93
SUB H	94
SUB IXh*	DD94
SUB IXl*	DD95
SUB IYh*	FD94
SUB IYl*	FD95
SUB L	95
SUB n	D6 n
XOR (HL)	AE
XOR (IX+d)	DDAE d
XOR (IY+d)	FDAE d
XOR A	AF
XOR B	A8
XOR C	A9
XOR D	AA
XOR E	AB
XOR H	AC
XOR IXh*	DDAC
XOR IXl*	DDAD
XOR IYh*	FDAC
XOR IYl*	FDAD
XOR L	AD
XOR n	EE n

Chapter 11

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

11.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose mark-up has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without mark-up, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

11.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

11.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete

Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

11.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled “Endorsements”. Such a section may not be included in the Modified Version.
- Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

11.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications”. You must delete all sections entitled “Endorsements.”

11.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

11.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

11.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

11.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.